TR den 10/165

Stellingen

behorend bij het proefschrift:

A Concept for Computer Aided Process Planning for Flexible Assembly

1. Het automatiseren van een complex informatieverwerkend proces als werkvoorbereiding is niet goed mogelijk zonder dat proces eerst te structureren. Een belangrijk inzicht dat daarbij kan ontstaan, is dat volledige automatisering onmogelijk, ongewenst of onnodig is.
2. Omdat het begrip *ontwerpen* reeds een zorgvuldige afweging van allerlei eisen impliceert, kunnen modekreten als DFA (Design For Assembly), DFM (Design for Manufacturing) en DFE (Design For Everything) heel goed vervangen worden door het simpele *Design*.
3. Gestructureerd ontwikkelen van software betekent voortdurend zoeken naar een optimum tussen een langere ontwerptijd en een verbeterde kwaliteit van de ontworpen programmatuur.
4. Een snellere terugkoppeling van (computerondersteunde) werkvoorbereiding naar produkt- en systeemontwerp kan de kwaliteit van de uiteindelijke ontwerpen aanzienlijk verhogen.
5. Het verdient aanbeveling om (in navolging van de gang van zaken bij NC-machines) vóór de aanschaf van een Industriële Robot bij verschillende leveranciers een applicatie gerichte afnametest uit te (laten) voeren: Dit geeft meer inzicht in de technische mogelijkheden van de machines dan de bijhorende folders; tevens geeft het een beeld van het vermogen en de bereidheid van de leveranciers tot het leveren van ondersteuning.
6. In een omgeving waar onderzoek en onderwijs faciliteiten delen, dient bij het vaststellen van de prioriteiten voor het gebruik van die faciliteiten niet alleen rekening te worden gehouden met het relatieve belang van de gebruikersgroepen, maar ook met de (on)voorspelbaarheid van het gebruik.

7. Praktische ervaring met ontwerpen en construeren is een essentieel onderdeel in het curriculum van een werktuigbouwkunde student. Het lijkt erop dat men dit bij de recente herstrukturingsoperaties uit het oog verloren heeft.
8. Naarmate de techniek voortschrijdt, begint het begrip 'elementaire montagehandeling' steeds complexer akties te omvatten.
9. De definitie van flexibel is flexibel.
10. Men krijgt meer waardering voor andersmans kookkunst als men zelf ook probeert te koken. In overdrachtelijke zin geldt hetzelfde voor het experimenteren met robots.
11. Topmanagers van bedrijven zouden zich vaker persoonlijk aansprakelijk moeten houden voor de invloed van hun bedrijfsaktiviteiten op het leefmilieu.
12. Onderzoekers zijn zowel wijs als dwaas: zij beantwoorden vele vragen maar roepen tevens nieuwe op. Onderzoek wordt hierdoor een zichzelf in stand houdend fenomeen.
13. Het grote aantal stellingen in proefschriften dat betrekking heeft op tekstverwerkers wordt niet alleen veroorzaakt door technische tekortkomingen van het door de promovendi gebruikte gereedschap, doch wordt tevens gevoed door frustratie met het eenduidig en duidelijk formuleren.
14. Op dit vel staan slechts dertien stellingen.

(Geïnspireerd op de paradox van Epimenides (een Kretenser):
Alle Kretenzers liegen)

50611
31757 24
7A 202.1016

TR diss
1816

A Concept for Computer Aided Process Planning for Flexible Assembly

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft
op gezag van de Rector Magnificus
Prof. drs. P.A. Schenck
in het openbaar te verdedigen
ten overstaan van een commissie
aangewezen door het College Van Dekanen
op dinsdag 17 mei 1990 te 14.00 uur

door:

Cornelis Jacobus Maria Heemskerk

geboren op 23 mei 1960 te Hillegom,

werktuigkundig ingenieur.

**Dit proefschrift is goedgekeurd door de promotor
Prof. ir. L.N. Reijers**

Heemskerk, Cornelis Jacobus Maria

A concept for computer aided process planning for flexible assembly

ISBN 90-370-0041X

©1990 by C.J.M. Heemskerk, Vogelenzang, Netherlands

Abstract

This thesis describes the development of a concept for computer aided process planning for flexible assembly. A hierarchical reference model for the assembly process in a single workstation is proposed. The assembly process is decomposed into four levels of abstraction. The highest level covers batch oriented assembly tasks that are decomposed via product and part level tasks into primitive level actions that are tightly coupled to physical devices.

A flexible planning approach is developed, creating and evaluating alternative strategies at each level of abstraction. The optimization of assembly plans was focussed on assembly cycle time. Heuristics are used in several stages to ensure that a good plan is found in a limited amount of time. At the product level, rough sequence plans are created directly from the product description. To reduce the complexity of assembly sequence planning a new technique for grouping parts (clustering) and a new method for representing assembly sequences (the Layered Assembly State Transition Diagram) are introduced. Stability rules further reduce the number of alternative sequences.

At the part level, a Part-script that explicitly describes the ordering of primitives guides the search process. Position uncertainties are propagated between primitive actions to verify that netto uncertainties in critical relations remain within pre-defined error bounds. When necessary, (position) sensor actions are automatically included.

To demonstrate the concept's feasibility, and to show that the chosen approach offers a generalized solution, a prototype planning system was built. This system (DAPPER, an acronym for "Delft Assembly Process Planner") generated process plans for different products on different assembly systems. In the future, integration of computer aided process planning with assembly

system and product design is foreseen.

Preface

This report is written in partial fulfillment of the requirements to obtain a Doctor's degree in the technical sciences from Delft University of Technology. Work on the project started directly after my graduation as a Mechanical Engineer in February 1985. The first year I stayed as a visiting scientist at the Robotics Institute of Carnegie Mellon University, Pittsburgh, USA. Through courses, laboratory experiments and personal contacts I had the opportunity to become familiar with the principles of Artificial Intelligence. I participated in the "Machinist Expert" project, an expert system approach to process planning for parts manufacturing.

Back in Delft, I joined the scientific staff of the Laboratory for Manufacturing Systems. From the first day on, many graduate students of the group were encouraged to participate in the project. Most of the experiments were carried out on test sites in the nearby Laboratory for Robotics and Flexible Automation (henceforth called the "Robotlab").

By the end of 1986, the project described in this thesis served as one of the focal points in the definition phase of the Delft Intelligent Assembly Cell (DIAC) project. DIAC was structured as a cooperative research effort with eight participating research groups from four faculties of the university. By the end of 1987, the DIAC project officially started and the project described here was completely integrated [Jon87],[Jon88]. Some topics that had been identified as interesting areas for further research in an early stage of the project (e.g. the user interface and the CAD system interface) receive relatively little attention in this thesis: they were made separate projects in DIAC. The experimental sites developed in this work serve as intermediate testbeds for DIAC.

Acknowledgements

I would like to thank all people and organizations that have supported me throughout the project. First of all, I would like to thank Nick Reijers, for providing me the opportunity to undertake the whole project, for his encouragement and support throughout the project.

Special thanks also go to Jan Markvoort of Philips CFT, whose advise has contributed to the structure of this thesis and to the approach of the project in general.

In the laboratory of Manufacturing Systems, Ton Storm provided general support and shared valuable insights when mechanical designs had to be detailed. Two other colleagues, Nico Boneschanscher and Hans van der Drift, participated in the project during their graduation work and continued to do so for some time before they started to work on their own PhD's. They contributed to the work and co-authored papers on assembly stability, clustering and position propagation. Peter Martens helped to define and experiment with the robot motion interface and provided support for the database management work.

Many thanks to the supporting staff in the Manufacturing Systems Laboratory and the Robotlab, for sharing their knowledge and helping out with experiments. I would like to mention especially Cui Qi, who encoded large parts of the planner, Guus Liqui Lung, who took care of much of the testbed engineering, and Joop Hoebeek, who provided PC and VAX support.

In the course of the project, many others have in some way been involved: Colleagues in Delft and at other universities, the users group from STW and a large number students (over 20 students participated in over 30 assignments). Through student assignments and robot programming experiments several companies were involved. It is impossible to mention everyone per-

sonally. Many thanks to all of you.

Financial support for the project came from the "Stichting voor de Technische Wetenschappen" (STW), the Ministry of Economic Affairs, Philips CFT, and the national SPIN/FLAIR program.

Finally, I would like to express my gratitude for the support of my wife Patty, for her understanding and encouragement, and for her participation in many long evenings of T_EX-hacking.

Contents

Abstract	i
Preface	iii
Acknowledgements	v
1 Introduction	1
1.1 What is assembly?	1
1.2 Assembly as part of the production process	3
1.2.1 A functional model	3
1.2.2 Trends in production	3
1.3 Flexibility	5
1.4 Flexible Assembly	7
1.5 Design For Assembly	9
1.6 Flexible Assembly Systems	10
1.7 Assembly Process Planning	15
1.7.1 A separate function	15
1.7.2 Goals	16
1.7.3 Overview	17
2 Planning	19

2.1	The evolution of CAPP	19
2.2	Trends and analogies	22
2.2.1	Developments in robot programming	22
2.2.2	Process planning in parts manufacturing	25
2.2.3	Planning in the context of AI	26
2.3	Problem decomposition	27
2.3.1	Robot oriented approach	27
2.3.2	A product oriented decomposition	28
2.4	Related research in assembly planning	33
2.5	Global Structure of the Planner	35
3	Batch Level	39
3.1	Plan optimization	39
3.2	Feeding strategies	41
3.3	Tool change strategies	44
3.4	Transport strategies	46
3.4.1	The IVF Mark II System	49
3.4.2	An implementation for manual assembly	50
3.5	The Batch level planner	50
4	Product Level: Plan representation	53
4.1	Assembly sequence planning: A combinatorial explosion . . .	53
4.2	Representing Sequence Plans	55
4.2.1	General Requirements	55
4.2.2	Assembly Sequence Planning Aspects	56
4.2.3	Scheduling	58
4.3	Candidate methods	59
4.3.1	Triangle Table	60

4.3.2	Precedence diagram	61
4.3.3	AND/OR Graph	62
4.3.4	Temporal Logic	63
4.3.5	Petri Nets	65
4.3.6	Constraint Networks	66
4.3.7	Evaluation	66
4.4	The Layered Assembly State Transition Diagram	71
4.4.1	Exploiting part redundancy	72
4.4.2	The ASTD	74
4.4.3	Sequence groups	77
4.4.4	A layered structure	78
4.5	The LASTD versus existing methods	85
5	Product Level: Sequence Planning	87
5.1	Structure of the Sequence Planner	87
5.1.1	The Environment	87
5.1.2	The Global Structure	88
5.1.3	The Input	88
5.1.4	The Output: The LASTD	94
5.2	Clusters	95
5.2.1	Definitions	95
5.2.2	Cluster Recognition	99
5.2.3	Theoretical implications of clustering	101
5.3	Stability heuristics	105
5.4	Expansion of sequences	106
5.5	Example: Cranfield Benchmark	108
6	Part Level	117

6.1	Part Level Planning	118
6.1.1	A script based approach	118
6.1.2	From script to plan	120
6.2	Geometric relations in an assembly environment	120
6.2.1	Existing approaches	122
6.2.2	The nominal position of an object	125
6.2.3	The Propagation of Spatial Uncertainties	126
6.2.4	Sensor interaction	129
6.3	Modelling the Assembly Environment	131
6.3.1	The nominal position	131
6.3.2	Uncertainty and Approximate Transforms	133
6.3.3	Example	136
6.4	Computing a critical relation	140
6.4.1	The basics of network reduction	140
6.4.2	Implementation	144
7	Primitive Level	151
7.1	Selection primitives	152
7.1.1	Select_feeder	152
7.1.2	Select_robot	153
7.1.3	Select_Grip	153
7.1.4	Select_sensor	156
7.2	Macro primitives	156
7.2.1	Part_feed	157
7.2.2	Part_grasp	157
7.2.3	Part_mount	157
7.2.4	Extended_line_scan	158
7.2.5	Three point measurement of part position	158

7.3	Low-level, device oriented primitives	160
7.3.1	Robot_move	160
7.3.2	Sensor actions	163
7.3.3	Tools	164
8	Experiments	165
8.1	Experiments on robot programming	166
8.1.1	Structured programming	166
8.1.2	A comparison of robot languages	169
8.1.3	DATE	170
8.2	Experiments on process planning and plan quality	172
8.3	Software Development	173
8.3.1	Approach	173
8.3.2	Database link	175
8.3.3	CAD-link	175
8.4	Testbed systems	176
8.4.1	The ASEA test site	176
8.4.2	The ROVIS site	178
8.5	Batch level experiments	182
8.5.1	Gripper exchange strategies	182
8.5.2	Sub-batch assembly with TES mounted grippers	184
8.5.3	Feeding strategies	184
8.5.4	Other system design influences	186
8.6	Product level experiments	187
8.6.1	The Cranfield Benchmark	188
8.6.2	Other products tested	191
8.7	Part level experiments	194
8.8	Primitive level experiments	198

8.8.1	Select_grip	198
8.8.2	Part feed on belt	199
8.9	Evaluation	200
8.9.1	Planning	200
8.9.2	Testbed engineering	204
9	Conclusions and recommendations	207
9.1	Assembly Process Planning	207
9.2	Testbed and prototype development	209
9.3	Recommendations	210
9.4	The future	212
	Samenvatting (Summary in Dutch)	227
	Glossary	232
	List of Figures	235
	List of Tables	235
A	Flexible Assembly Systems in Literature	237
B	Dataflow Diagrams and Data Tree	241
B.1	How to read a ProMod dataflow diagram	241
B.2	Dataflow Diagrams	243
B.3	Data Tree	243
C	Timing tables of batch level experiments	263
C.1	Multigripper kit assembly	264
C.2	Kit assembly with TES mounted grippers	264
C.3	Sub-batch assembly with TES mounted grippers	266
D	Implemented primitives	276

Chapter 1

Introduction

1.1 What is assembly?

The term assembly can either be used to refer to a process of bringing together parts to form a unit, or to the result of that process (a product). Here the term will be used for both purposes. In everyday conversation, the assembly process is thought to cover only part handling and joining operations. In this thesis, as usual in the context of automatic assembly systems, the term will be used in a broader sense, also covering pre- and post-processing operations like cleaning, adjusting and inspection [Gai81], [RdH86].

Central operations in assembly are handling and joining.

Handling covers operations like recognition, separation, grasping and positioning of parts, in other words, operations that alter the geometric situation of parts with respect to other parts, without altering their shape or composition.

Joining means bringing parts together, reducing their relative degrees of freedom, and establishing a more or less fixed connection between them. In the German language literature extensive classifications of joining methods are available in [Rot80], [VDI75] and [DIN67] among others. A brief overview of joining methods inspired on these classifications:

- Join by bringing together (e.g. insert, hook on)

- Fill (with liquid)
- Join by force (e.g. screw connections are partially established by force)
- Join by shaping (e.g. outsert moulding)
- Join by plastic deformation (e.g. bend around, staple)
- Weld
- Solder
- Glue

An important factor from a functional point of view, is the "connectivity" of the joining method: How many (relative) degrees of freedom (DOF) are left after assembly, and is it possible to take the product apart again without causing damage?

In the work described in this thesis the emphasis was on "basic assembly": Parts handling and reversible joining methods (joining by bringing together and joining by force). Bringing parts together is a common element for all assembly operations. Screwing is the one joining method that still dominates in industry, although in some cases screws are being replaced by snap fits.

Welding, soldering and glueing are processes that establish a material bond between parts after they are brought close to one another. The first part of the operation (bringing the parts together) is basic assembly, the second part establishes a connection that completely fixes the parts with respect to another (no relative DOF are left) and is usually irreversible. Shaping or plastic deformation also result in irreversible connections, but limited degrees of freedom may be possible (e.g. rotation around a rivet).

From a production point of view (the number and type of assembly operations involved), it is important whether a connection involves extra elements (e.g. screws, glue) and whether special tools are used (e.g. a welding gun). Here we find a second reason to exclude joining methods other than bringing together and joining by force: these processes are complete topics on there own, requiring special tools and process expertise that goes beyond the scope of thesis.

Finally, the work described in this thesis was restricted to assembly operations that involve parts with a predefined, constant shape, excluding joining

by filling and assembly of flexible parts. Again these are topics in their own right.

1.2 Assembly as part of the production process

1.2.1 A functional model

Fig. 1.1 shows the assembly process as a function in the context of a (general) production company. The major task of the assembly process is to transform loose parts and subassemblies into assembled products.

For its control information, the assembly process relies on the interaction with process planning and production control. Also there is an interaction with a materials management system and a little explored link to quality control.

1.2.2 Trends in production

In the industrialized countries one can recognize the following tendencies in production:

- Customers are becoming more sensitive to new products, requiring increased frequency and speed of product development. Product market life becomes shorter. Also, there is a growing demand for customized products. Because the total volume of production only grows slowly, these developments mean that the spectrum of product types in production at one time (the product mix) will cover more products and product variants.
- Customers put increasing demands on product quality, which has to be high and uniform. They require better products, and increased reliability in delivery.
- Increasing competition forces companies to put increasing demands on the product design, thereby making it more important to rationalize and optimize production in all stages.

The market developments described above require production companies to increase their ability to adapt their production processes quickly to changes

Figure 1.1: Functional Model of a Production Company

in the market demand. The key concept to a fast response is flexibility.

1.3 Flexibility

Flexibility is the ability to adapt to changing circumstances. The main goal for any (flexible) production company is to make those products the customers ask for, when they ask for them, with the right quality, in the right quantity and at the right price. Therefore, the best definition of flexibility is one that describes the ability of a production system to react to changes in the market.

Flexibility comes in many ways, and can e.g. be defined in relation to the frequency with which adaptations are made, in relation to the nature of the changes, etc. Here, flexibility is related to the life-cycle of a production system. Based on ideas provided by Andreassen [AA88] and Fabricius [Fab85], five types of flexibility are distinguished (fig. 1.2):

1. Planning flexibility is a system's or system component's ability to be adapted to a broad range of tasks, for the first time during the first planning stages, but repeatedly when the system is dismantled and parts of it are re-used to build a new system. Reconfiguration of a system can also be necessary to make it more suited to a different mix of products. High planning flexibility reduces the risk of investment. It can be attained by using modular units and standardized interfaces, both in a mechanical sense and with respect to control, and by giving the system a flexible layout.

2. Running-in flexibility is the ability of a system to be easily adapted and fine-tuned to a specific task during the early stages of system life (the set-up phase). High running-in flexibility assures that the system's capabilities can be rapidly verified and that the system will be running with the required capacity within a short time after configuring has begun, saving time and cost, and thereby reducing the risk of investment. It can be reached by using proven technology, duplication of stations, using standardized interfaces, and well-thought-out adjustment and calibration procedures.

3. Insensitivity is a system's ability to deal with (minor) variations in the process, caused by deviations in parts or system components. High insensitivity means that the system has a high availability and reduced man-power requirements. It can be reached by using fault tolerant equipment, creating simple exception handling routines (usually a combination of "smart"

Figure 1.2: Five types of flexibility playing a role in the life cycle of a production system

preprogrammed strategies and flexible equipment), using component part feeding strategies with built-in redundancy etc.

4. Versatility is a system's ability to change over or be changed over quickly and easily to a new task that was foreseen during system development (e.g. a new product that is a member of the current product family). High versatility means a gain in productivity due to short change-over times. It can be reached by using universal equipment, tools and methods (e.g. a robot, sensors, a structured programming environment), or equipment and tools that can be easily exchanged (e.g. an automatic tool changer, a multi-gripper).

5. Adaptability is a system's ability to change over or be changed over quickly and easily to a task that was not known or not given high priority during system development, related to a new or strongly modified product family. High adaptability in the first place means a reduced investment risk, but it also creates a greater preparedness for new market developments. It can be reached by using modular system components with standardized interfaces, both in hardware and software, well structured programming environments for programmable equipment, and special tools like fixtures that are easily adjustable or consist of a (potentially expensive) generic part combined with low-cost product specific parts.

A company that wishes to achieve flexibility towards its market, has to adjust the flexibility of its production system to the variance in the product design. This dissertation will concentrate on a part of the production process that has not been rationalized as much as the other parts: *Flexible Assembly*.

1.4 Flexible Assembly

Changes in the area of flexible assembly can be predicted by analyzing the general market developments, and by drawing rough parallels with other production processes:

- Products are made in more variants. The total volume of products does also increase but not as rapidly. Therefore, the production mix will grow. Also, product life becomes shorter. Both developments result in a growing need for new process plans.
- The definition of a product family will widen (a situation comparable to parts manufacturing, where machining centers can produce parts

Figure 1.3: *Application of Industrial Robots in Germany (Source: IPA)*

in a large range of sizes, only limited by the machine's working range, the number and configuration of its axes and the readily available set of tools). Production systems have to support this development by increased versatility.

- Changes to the product mix occur on shorter terms, requiring a quick system reconfiguration capability.
- The total production process is being rationalized. Following improvements in the manufacturing process and product design, the assembly process cannot escape rationalization. The application of flexible assembly systems is increasing and is expected to increase further [ABW84], [NN88]. The rapidly growing number of industrial robots employed in assembly (figure 1.3 shows the German situation) supports this expectation.

The conclusion is that there is a need for increased flexibility in automatic assembly. Increased flexibility can only be achieved through a combination of a flexible product (family) design, a flexible system to assemble the products and a flexible method of process planning. The work described in this dissertation concentrates on the latter, and concerns the development of a

flexible, computer supported method of process planning that exploits the flexibility of the assembly system and systematically explores (assembly) action alternatives allowed by the product design.

1.5 Design For Assembly

The first precondition to achieve a flexible, automated assembly process, is that the product and its family have to be designed with the principles of Design For Assembly (DFA) in mind. The product design should evolve from a family concept, that specifies all variants, variations in components and as far as possible, also future variations.

Literature provides many descriptions of DFA techniques, see e.g. [BD79], [AKL83], [Bas88], [Baa88]. Most methods have primarily a relative value: they serve to compare alternative product designs. Suggestions for design changes come from design rules or catalogues with examples. There are two basic principles: *Eliminate steps from the assembly process* (e.g. by eliminating a part) and *Make assembly simpler* (e.g. by giving all parts the same assembly direction).

In the Laboratory of Manufacturing Systems, there is some experience with a method based on the work of Boothroyd [BN83]. Results indicate that the application of DFA techniques can substantially decrease assembly cost, among others by eliminating parts [Jam88], [Mar88b]. However, the same design changes lead to even more substantial savings in manufacturing and inventory. On the other hand, some parts may become more complex, making them more expensive. The conclusion is that DFA should be placed in a broader context of Design For (Flexible) Production (see also [SHM89]).

Earlier work in the laboratory showed that not being able to adapt a product design in ways suggested by DFA analysis, will hinder the application of Flexible Assembly Systems [Ver85], [Rei89]. The application of DFA techniques is most rewarding when a completely new product family is designed.

In this thesis it is assumed that DFA principles have been applied during product design. At certain points it will be shown that process planning is supported (made easier) by structured product design.

1.6 Flexible Assembly Systems

The second precondition to achieve a flexible assembly process is that the assembly system has to be designed as part of a larger flexible production system. First of all, rationalization in previous production stages like parts manufacturing is assumed to have occurred. Furthermore, the assembly system itself has to be designed with flexibility in mind. The assembly system should be designed as a whole. Not only a core element like a robot has to be flexible, but also the feeding equipment, the product transport system etc. A flexible system will use modular components and standard interfaces.

Characteristics of Assembly Systems in Industry and Research

To obtain an impression of the flexibility that is achieved in existing assembly systems, we made a review of literature, covering about 50 systems in use in industry and under development in research laboratories [Goo87]. An overview is presented in appendix A. The study identified the following characteristic factors:

- The complexity of product and family, depending on:
 - The number of parts per product
 - Number of different parts per product (number of part types)
 - The number of product variants in the family
 - Developments in the product mix (the number of new variants per year)
- The size of the production problem, as indicated by the total volume of production and the typical batch size.
- The complexity and flexibility of the assembly system, depending on:
 - The number of robot arms in the system
 - The number of different parts (part types) handled by one arm

Some conclusions based on the literature review:

- FAS of the current generation are restricted to one family of products. In the future, a development towards a wider definition of the family

concept is envisioned [AG87], [AA88]. This development however, is not expected to result in the ultimate "*Universal Assembly Machine*". Instead, the flexibility of future assembly systems will be tailored to expected changes in the market, e.g. with quick reconfiguration capabilities in systems for consumer goods (with a short market life) or high versatility in systems for products with a longer life-cycle but large changes in the composition of the product mix.

Figure 1.4: Yearly production volume

- Production volumes (estimated from the product cycle time) are roughly between 100,000 and 1 Million products per year (see fig. 1.4), corresponding well to general economical indications by Boothroyd [BPM82] and De Winter [vBdW87] among others. Exceptions on the high volume side are small products with only a small number of parts (e.g. lipstick, watch casing), exceptions on the low volume side are large and/or complex products (e.g. cylinder head, lawn mower).
- The number of parts in a product shows a wide variation. The "average" product contains about 25 parts (fig. 1.5). Among the exceptional

cases with (many) more parts are cylinder heads, a keyboard (cap assembly) and PCB's, involving many relatively small parts and a high repetition factor. Examples of products with a small number of parts are a ball-point pen and a fuel gauge, relatively simple products that are assembled in large quantities.

Figure 1.5: *Total number of parts per product*

- For the average product, there are half as many unique part ID's as there are parts (see fig. 1.6). Usually there are some unique parts, other parts (e.g. standard fasteners) re-occur more often. Products with a small number of parts in total tend to have more unique parts, products consisting of many parts tend to have a higher repetition factor.
- The average assembly time per part per robot arm is about 12 seconds for cell structured systems, and about 6 seconds for line structured systems. Factors with a large influence on these times are part size, part weight and complexity of the assembly motion. A typical example of a slow process is car windscreen assembly (one system had a cycle time of 1.5 minutes).
- The average robot arm handles a total of about 8 parts, one typical

Figure 1.6: *Number of part types per product*

exception being a robot that assembles 55 keyboard caps.

- One robot arm never handles more than 10 different part types, robots in line structured systems often handle no more than 1 part type (fig. 1.7).
- The versatility and adaptability of the systems is limited because much of the peripheral equipment is part specific.

Literature indicates that assembly systems with more flexibility than that of the current generation, including more flexible tools and peripheral equipment, are indeed feasible and capable of assembling several product types in one system [AG88], [vBdWT*86], [SFD87]. Successful development of flexible tool prototypes in the Laboratory for Manufacturing Systems further supports this assumption (see chapter 8).

Another development is that assembly robots tend to get faster. This means that the typical cycle time will become smaller and economic feasibility of automatic assembly with robots improves.

Figure 1.7: *Number of part types per arm*

1.7 Assembly Process Planning

1.7.1 A separate function

A standard approach to get an impression of a new system or process is to create a black box around it and examine its in- and outputs. An intuitive black model of the assembly process covers the transformation of a group of loose parts and subassemblies into an assembled product (fig. 1.8).

Figure 1.8: *The assembly process as a black box*

The interaction between process planning and the actual assembly process takes place inside the black box. This model looks attractive, hiding detailed knowledge about the assembly process in the system itself. An intelligent controller inside the box should be able to infer the best way to proceed, using only a description of the desired product and its own process knowledge. In theory this could result in a assembly system that would respond quickly to changing circumstances. There are several reasons why I think process planning should be treated as a separate function.

There is a growing need for more flexible process planning. The need will occur both in "small" series assembly with a limited number of relatively complex stations, as well as in large series assembly with a large number of relatively simple stations. One area of interest is faster program generation for robots and peripheral equipment. Assuming that automatic assembly can be justified, small series assembly then implies an elaborate product mix for which many relatively complex programs have to be generated. In the second case a large number of stations has to be reprogrammed, and

although the task of each station might be relatively simple, the *number* of stations and more frequent changes in the product mix make fast reprogramming important.

Faster feedback from process planning can also improve system and product design: During assembly process planning the consequences of the decisions made during product and system design become clear, because this is the stage where actions that will put the product together are assigned to specific devices in the assembly system. Product design e.g. influences the "graspability" of a part and the stability of a subassembly, while the system design e.g. influences robot motion time by constraining the distance from feeder to assembly fixture.

A problem is that assembly process planning is difficult: It requires extensive knowledge of the process and of capabilities of tools that, except for simple products, is hard to capture. The process planner has to optimize his plans by choosing a system layout, the tools he will use, the sequence of actions for each product and each part etc. Most of this process planning is at present done manually and making a good plan is tedious, time consuming, and therefore expensive. The planner will try to do this only once, and use the same plan over and over again.

The assembly tasks typically performed in automated assembly systems have a repetitive nature because products are made in larger quantities. The tasks themselves are also similar from one product type to another. If the information needed to generate programs for the systems' components could be made accessible in some digital form (e.g. from a CAD database), it is conceivable that mechanisms for computer supported program generation could exploit the repetitive nature of the assembly process and largely reduce the efforts needed for (re)programming.

1.7.2 Goals

The previous paragraphs set the background for the research described in this thesis. The main goal was to develop a new concept for computer aided process planning for flexible assembly and to demonstrate the feasibility of this concept. More specifically:

1. To develop a new concept for a computer aided process planning system for flexible assembly systems, that can reduce the effort needed

to generate assembly programs and provide faster feedback to system and product designer. The concept should be generally applicable (not limited to one specific product or system), and extendable, so that new assembly techniques, system components or product design can be incorporated.

2. To create a reference model that can be used to describe the assembly process in terms of goals and intermediate states, such that (partially) automated reasoning becomes possible.
3. To create and develop data structures and algorithms that can be used for automated reasoning. Potential sources of information (e.g. CAD-system, user interface) should be examined. Parts of the reasoning process that are difficult to automate within the near future should be identified.
4. To build a prototype process planner and a prototype assembly system that demonstrate the feasibility of (partially) automatic plan generation.

The optimal assembly plan simultaneously satisfies many goals. Given the scope of this project, it was not feasible to address all factors that can be optimized. This thesis focuses on minimization of the assembly cycle time and improving plan robustness by increasing plan flexibility (plans should be easily extendable, or include alternative strategies).

Potential strategies to minimize the total time needed to assemble a complete product are elimination of redundant actions, reduction of the time needed for essential actions, and making actions concurrent. The planner should exploit possibilities provided by product features and system characteristics.

In accordance with the main line of assembly research in the Laboratory (small series assembly), the attention was focused on the assembly process in a single workstation with a single assembly robot. Several other restrictions were assumed, as described in section 2.5.

1.7.3 Overview

The next chapters will go deeper into various aspects of assembly process planning. Chapter 2 will compare developments in assembly process planning with developments in process planning for parts manufacturing, present

a new, hierarchical reference model for assembly process planning and describe the context and global structure of an intelligent process planning system.

Chapters 3 to 7 will go successively deeper into each layer of the hierarchy. Chapter 3 starts with the highest level: the Batch level. Chapters 4 and 5 deal with the product level, including the generation of assembly sequences. A new technique for grouping parts to simplify sequence planning and a new method to represent assembly sequences are introduced. Chapter 6 deals with assembly actions at the part level, where accumulation of uncertainties and alignment problems arising from the chaining of several primitive assembly actions are solved by the selection of sensors or special tools.

At the primitive level described in chapter 7 the final links to the devices that will execute the assembly actions are established. Issues like the selection of a gripper or planning the shortest collision free path are discussed.

Chapter 8 describes the development of two prototype systems and the prototype products that were used to verify the ideas developed during this project. The development of software for the planner, scheduler and dispatcher and the procedures for the experiments are discussed, and the results of the experiments are evaluated.

Chapter 9 finally presents the main conclusions and gives recommendations for future work.

Chapter 2

Planning

2.1 The evolution of CAPP

In the application of computers to support manufacturing process planning there has been a development from a variant based approach (elaborating pre-defined plans) to a generative approach (creating new plans from scratch). This development should be seen in relation to the evolution of the manufacturing industry (see table 2.1). In the present situation, the production industry is information intensive and automation is concentrated in "islands". The challenge of the future is to achieve a higher degree of integration (Computer Integrating Manufacturing or CIM). Computer Aided Process Planning (CAPP) is seen as a critical link between design (CAD) and manufacturing (CAM), and thus as a critical step in achieving CIM [HL88a].

Early work in CAPP used the variant approach, and was predominantly based on Group-Technology (GT). Parts having similar features are grouped into part families, and a unique code is generated for each family. For each family, a standard process plan is generated and stored in a computer. The GT code is used to identify new parts and retrieve related process plans. Usually for one part or family, a plan is generated only once.

The second generation of computer-aided process planning systems were based on a semi-generative approach. After a part and its family are identified, there are several options. The first is to adapt the standardized family process plan. A second option is elaborate on a basic plan. A third option is

to create a completely new process plan from a set of standard (predefined) process descriptions.

Stage of Evolution	Character of Industry	Type of Automation	Type of Planning	Key Technology
Past	Labor Intensive	None	Manual Planning	
Past	Equipment Intensive	Separated Automation	Computer Aided Retrieval	Group technology
Present	Information Intensive	Interfaced Automation	Semi-generative Automatic Planning	CAD Representation
Future	Knowledge Intensive	Integrated Automation	Fully-generative Automatic Planning	Artificial Intelligence

Table 2.1: *The evolution of CAPP Methods (adapted from [HL88a])*

The current emphasis is on creating fully generative process planning systems. First of all, a link to a CAD system has to be established. One problem here is that current CAD systems do not completely specify the product geometry and topology in a structured manner. Secondly, early generative CAPP systems used decision tables and decision trees to capture manufacturing logic. These methods are not flexible enough. Representation concepts from knowledge based systems and AI-based planning techniques have to be incorporated.

To some extent automatic (generative) plan generation has already been realised: In the electronics industry, the design of a Printed Circuit Board on a CAD system can be automatically translated to a pick-and-place program for a component placement machine. This approach only works for standard components like resistors and IC's and because the PCB assembly process is conceptually very simple (only vertical insertion, virtually no sequence restrictions).

For mechanical and electromechanical products, on which the work described in this thesis focuses, the assembly process is much more complex. Completely automatic planners are not expected within the next decade. Still, increased flexibility in process planning is seen as one of the keys to a more flexible assembly process (see Chapter 1).

The goal of the work described here is to increase the understanding of CAPP for flexible assembly. A groundwork for an automated, generative process planning system will be provided, and a prototype of such a system (called "Planner") will be built. The emphasis of the work is on the reasoning that takes place during process planning, and on modelling the assembly process at various levels of abstraction.

The approach of the work is automation oriented (i.e. it is attempted to generate process plans completely automatic). This does *not* imply that fully automated process planning is seen as the only or ideal future. The main reason is that automation enforces a very structured approach. It is expected that attempts to automate (parts of) the planning process will clearly reveal the areas where planning is complicated. Similar results could also be obtained via a completely interactive, user-oriented approach. Practical planning systems for the next decade will have to combine the best of the two approaches.

To demonstrate the feasibility of the approach, the Planner prototype resulting from the project should be able to produce automatically assembly plans for simple products. Its structure should be open, so that later it can serve as the basis for a more powerful automatic system, or as the basis for an interactive tool. Such an interactive system might work in several modes: it might methodically compare process alternatives proposed by a human Planner or, the other way around, it might propose rough plans that can be further refined by a human.

First, the current situation in assembly process planning is analysed. Analogies are drawn with developments in robot programming, with process planning for parts manufacturing and with planning in an AI context. Then the environment of an assembly process planner is briefly discussed, followed by the description of a new reference model that divides assembly process planning in four layers of abstraction.

2.2 Trends and analogies

2.2.1 Developments in robot programming

Off-line programming

In Flexible Assembly Systems currently in use in industry, substantial parts of the overall process plan are encoded in the form of robot programs. Developments in robot programming are thus expected to be reflected in developments in process planning for assembly. A first interesting development is that of the robot programming approach. According to the use of the target robot during program development, approaches can be classified in on-line, hybrid, and off-line programming.

During on-line robot programming, the robot is physically guided through the motions. The oldest existing on-line method (still used e.g. for spray painting robots) is playback programming, where the robot is literally taken by the hand. The robot controller carefully records all motions in the teach phase and exactly replicates them during program execution. Playback programming in its pure form does not allow sensor interaction and there is no language to support conditional actions or subprograms. Therefore, playback programming is less suited for flexible assembly.

A more modern on-line approach is teach-in programming, where the robot is taught points instead of paths. The robot is usually not directly manually positioned, but through a push-button controller or a joystick. The exact path the manipulator will take between the points during program execution can be unpredictable when Point To Point (PTP) control is used. Most recent robots models can however be equipped with a (more expensive) Continuous Path (CP) controller, that allows the user to specify paths explicitly (e.g. being a straight line or a circle). Teach-in programming is used for most of today's robots. One problem that becomes larger when there are many new programs to be generated, is that throughout the program generation phase, the robot cannot be used for production purposes.

For hybrid program generation, programs are divided in a control flow part and a position data part. The control flow part is programmed off-line, position data is either left open or just referred to by variables. The exact positions are taught on-line and inserted automatically into the empty slots in the program or dumped into a separate coordinate file that is linked in

with the program later. Unfortunately, the on-line part is still rather time consuming.

When the off-line approach is used, both program control flow and position data are generated off-line, without the physical manipulator having to be present. An important precondition is that there has to be some (geometric and kinematic) model of the robot and its environment available. Commercial off-line programming systems of the current generation are mainly geometry oriented. They have a library of predefined robots and the environment model is defined interactively in a way similar to a CAD system (or an interface to a commercial CAD system is provided). A very important feature is the visual feedback via a graphical display. Experience in the WPS laboratory with "off-line" robot programming systems indicates that systems that do not include visual feedback should be regarded with skepticism.

Examples of commercial off-line programming systems are MRS (from McDonnell Douglas), ROBCAD (from Technomatix) and GRASP (originally developed by the University of Nottingham UK, now being marketed by BYG) [BDTG83], [Goo89]. These systems are relatively expensive (100,000 – 200,000 Dutch guilder excluding hardware) and only several hundreds of systems have been installed world-wide. They are mainly used for off-line programming of robots in car manufacturing (spot welding) and to simulate the robot - environment interaction in geometrically complex situations, in order to optimize the system layout before it is actually built.

The way in which robot motions are specified in such a system can be described as "off-line teaching": The user specifies (although more indirectly) every motion of the manipulator arm. As will be shown later, this off-line specification of robot motions is an essential ingredient of assembly process planning, but the latter involves much more than that.

In the context of the work described in this thesis, a current generation off-line robot programming system is regarded as a potentially powerful element of a generative process planning system. It could be a tool to implement a graphics oriented user interface. It could also provide visual feedback and support reasoning with kinematic chains.

A thorough evaluation of one system (GRASP) was made [Goo89]. The system proved its value in a layout study for a handling application. The visual feedback and the cycle time estimates provided by the system could be used to create an improved layout (putting the same elements in a differ-

ent configuration), and supported the selection of an optimal process cycle. Simulation showed a 12% reduction of the cycle time. Automatic detection of collisions helped to improve end-effector design.

On the negative side, interaction with the environment was hard to model (only binary sensors could be simulated, no PLC drivers were available). Information on the internals of this system, another precondition for successful integration with the higher planning levels, proved to be difficult to obtain. System cost was yet another restriction.

New systems under development, e.g. ROSI (under development at the University of Karlsruhe, [DH85]), USIS (a project of the University of Munich, [BHMS89]) and SMS (developed by Siemens AG, [Goo89]) also start from geometry and kinematics, but aim for a more open approach and offer interfaces for robot dynamics, sensory control, and databases to maintain object relations etc. The main problems of these new systems are that they are still under development and that there are only a few (robot-specific) drivers available. In the work described in this thesis, integration of an off-line robot programming system with the Planner was not further pursued.

Programming languages

A second interesting development in robot programming is that of the robot programming languages [Loz82], [BS82], [Hoc86], [Rem87]. In general this development shows a parallel with general computer programming languages with a 5–10 year time lag.

Languages for commercial robots tend to be more "backward" than languages developed in research environments. Early commercial languages like VAL [Shi76], [BJ83b] could be compared to FORTRAN or BASIC, later (e.g. in AML) structured programming as in ALGOL or PASCAL was introduced. The newest languages (e.g. IBM's AML2) include limited list processing and object oriented features, clearly inspired by AI oriented languages like LISP and Smalltalk. This trend is expected to be reflected in process planning, where AI oriented languages gain influence and AI-techniques will get more emphasis.

The top-of-the-line robot languages have evolved from manipulator control to workstation control languages, and robot controllers have become complex multi-tasking systems (e.g. the VAL+ system from ADEPT). Sometimes the controller even includes a separate PLC for synchronization and

control of peripheral devices (an example is the BOSCH Rho2 controller, a limitation of this system is that off-line programming of the PLC and the robot is not integrated).

Commercial robot languages have the problem that they are robot or supplier specific. There are some standardization efforts underway that might result in a "standardized" robot language with limited functionality. This standardized robot language will probably be implemented as a driver or subroutine package augmenting a general programming language [HP86]. The driver software will either be running on a simplified robot controller or as one task in a multi-tasking system. The workstation controller could run on the same system.

Robot languages in research have been growing from structured, explicit languages (e.g. AL [BJ83b], LM [LM81], SRL [BJ83a]) to more implicit, object- or task-level languages (e.g. LAMA [Loz76], AUTOPASS [LW77], and RAPT [PAB78]). The latter languages have a built-in way of referring to the geometry and features of objects and include (or hide) a substantial part of assembly process planning in their specification and will be treated in more detail in chapters 4 and 5.

In the work described here, both the standardization route and the post-processor approach are explored. In one implementation, a robot is treated as a relatively simple "motion interpreter", getting its assignments through a standardized on-line interface with limited functionality. In the second implementation, motion commands are translated by a post-processor into a program in the robot's off-line language (see chapter 8).

2.2.2 Process planning in parts manufacturing

Process planning (PP) for parts manufacturing has many parallels with process planning for assembly. It also concerns operation sequencing, selection and detailed specification of tools, fixtures and processes that establish a product transformation. A difference is that PP in parts manufacturing is related to one specific part, whereas assembly deals with many inter-related parts.

CAPP in parts manufacturing is several years ahead of CAPP in assembly. Available systems cover the whole spectrum from variant based planning to fully generative planning. As predictable, process planning for conceptually relative simple processes like turning is further ahead (MITURN, ROUND).

At the 1986 CIRP Round Table, prominent production engineering scientists [D*86] recognised the need and the possible advantages of using AI techniques in Manufacturing Engineering. CAPP was identified as one of the major application areas.

One popular AI-technique is the expert systems approach. Heuristics in the form of rules are used to capture process planning knowledge. Examples of systems using this approach are GARI [DL81] and the Machinist Expert Project of Carnegie Mellon University, in which the author participated [Hay87], [Bou87b], [HW86], [WB88], [Hee86]. Another example is the XPLANE system developed at the University of Twente [vtEK86], [vtE88]. It is expected that other AI-techniques will also find application.

Another development in parts manufacturing is increasing integration. Heuristic knowledge is expected to cover only part of the problem. Integration of systems that cover mechanical and geometric details of operations (e.g. Finite Element Modellers) and integrating of feedback on results of previous plans is necessary to achieve improved plan quality.

Process planning will be integrated with product design on the one hand, and with production control on the other. Expected results are a.o. improved efficiency in the information flow, improved quality of plans by reducing human errors and by providing faster process feedback to the design. Increased flexibility will result in an overall higher system productivity.

Expected problem areas in achieving higher integration are expected to be similar in process planning for assembly: The development of consistent product models, uniform database and user interfaces and closing the feedback loop with process experience.

2.2.3 Planning in the context of AI

Generative process planning for flexible assembly has many aspects in common with what is called planning in the world of Artificial Intelligence. In a generic sense, planning is viewed as the activity of developing a way to achieve a desired goal under given constraints and with limited resources. Key elements are goals, constraints and resources. Planning covers a multitude of activities, from problem-solving via constraint-propagation to resource optimization.

The work described in this thesis focuses on the mechanics of planning,

that is, on the development of algorithms that exhibit intelligent planning behaviour when run on a computer. The nucleus of planning is a search problem that can be broken down into three basic steps:

1. Finding a good representation for the problem, including a clear specification of the start and goal situations and relevant constraints.
2. Generating alternative solutions ("filling in" the representation scheme), by selecting resources to perform actions encoded in the representation.
3. Searching the representation for an optimal solution (the optimal path from start to goal).

An AI-approach was used to formalize the assembly process planning problem. Goals and intermediate states had to be described explicitly. Assembly operations that influence planning decisions had to be identified and expressed in a formal way. The effect of each planning step had to be expressed in terms of a distance to the goal.

In this formalization phase it became clear that the assembly process planning problem as a whole is too large to be handled in one time, and that should be decomposed into smaller (sub)problems. AI-literature suggested that a hierarchical decomposition could be a useful approach. This is the topic of the next section.

AI-techniques were further of help in solving the planning (sub)problems that were formalized. The emphasis has been on applying algorithms and approaches that have been proven to lead to acceptable solutions (e.g. heuristic search).

2.3 Problem decomposition

2.3.1 Robot oriented approach

A first approach considered for the decomposition of the assembly process planning problem was to focus on the robots in the system and treat assembly process planning as a special application of robot programming.

Robot programming is usually split up into four levels, according to the way the robot actuators get their commands: servo level, robot or manipulator

level, object level and task level [Loz82]. Only the task level is considered to be independent of the robots that eventually will perform the assembly task. For its knowledge, a task level programming system relies on a very detailed description of the environment, called a World Model.

The level of the languages currently available for industrial use lies somewhere between manipulator and object level: modern languages like AML2 give enough flexibility and possibilities for interaction with external equipment to allow object level programs to be written. However, the user still has to supply all the building blocks himself (like exactly how to "get" an object). Also robot motions still have to be specified explicitly. The assembly programs generated in this way will depend very much on the type of robot used. Higher level concepts e.g. relating to the composition of products or product transport are not supported.

2.3.2 A product oriented decomposition

Here, a more product oriented decomposition is proposed. The basis is a reference model for assembly process planning with four abstraction levels: batch level, product level, part level and primitive level (fig. 2.1).

At each level, planning decisions are taken in two phases. In the first phase the assembly process is analyzed in a product (or task) oriented way (as independent of any specific assembly system as possible). In the second phase the applicability of specific elements in the assembly system to the assembly tasks at hand is examined.

The four abstraction levels will now be discussed briefly. They will be treated more in depth in chapters 3-7.

Batch level

At the Batch level, a single batch of products is considered. The process planning approach described in this thesis was developed assuming that the batch contains only one type of products, although extension to mixed batch production seems feasible. The batch level was chosen as the upper boundary of the process planning problem mainly for practical reasons. The batch level represents the lowest level at which one would make tools available to an assembly workstation (one does not change the configuration for just one product).

Figure 2.1: A product oriented decomposition of the assembly process into four levels of abstraction

Decisions on the batch level concern the availability of tools and global strategies of assembling a batch of products in one workstation. There is a strong interaction with assembly system layout design. The influence of decisions made during system design is described in chapter 3. An experimental evaluation of some representative tool exchange parts feeding and pallet transport strategies is offered in chapter 8.

Product level

Figure 2.2: The assembly process as a sequence of join operations. On the left a linear sequence, on the right a network structure, representing a set of three sequence alternatives

The product level deals with only one product at a time. At this level the assembly process is regarded as a series of join operations on parts and subassemblies. The sequence in which these join operations should take place

can e.g. be represented by a linear sequence (fig. 2.2-a) or by a network of actions, where the branches represent alternative sequences (fig. 2.2-b). In the start state all parts are "loose", and in the final state (the goal state) all parts are "joined".

Apart from the method used in fig. 2.2, there are many other ways to represent assembly sequences. An overview of various existing methods and the development of a new method are presented in chapter 4.

During the first phase, the product is analyzed to see in which order the parts can be treated. The emphasis is on constraints that are caused by the structure of the product (collisions between parts during assembly).

In the second phase the influence of tools like fixtures and special grippers is examined, e.g. whether a subassembly is stable enough to be "allowed" as an intermediate step in building the product, or whether parallel handling of multiple parts is supported.

A product level assembly plan should at any moment during the assembly process indicate the best next step, i.e. which part should be assembled next. For reasons of flexibility, the ideal plan should leave some sequencing options open. Chapter 5 describes how sequence plans can be generated.

Part level

At the part level only one part at a time is considered. From the parts' point of view the assembly task at this level is basically a fixed sequence of actions, bringing it from the "loose" to the "joined" state (fig. 2.3):

The challenge at the part level is not so much in finding the right sequence of actions for each part, but in picking the best strategy for each step and in keeping track of the relations between the objects in the assembly environment (e.g. robot, grippers, parts, feeders).

A step at the part level is successful if there is enough information about, or control over the part, to perform the next step. Each step should result in more information about or more control over a certain part. A goal of the assembly process is to establish specific geometrical relations between previously unrelated parts. It is a responsibility of the part level to make sure that a part creates the right relations. Part level planning is therefore tightly coupled with representing and reasoning about geometric relations.

Figure 2.3: *Elementary sequence of actions at the part level*

The propagation of position information between the steps plays an important role. Whereas the details of exactly how to gather position information are left to the primitive level, the part level deals with the cost and speed of acquisition and the reliability of the information needed to assemble the part successfully. When there is not enough information available a sensor action can be added. Chapter 6 describes how the assembly environment can be modeled so that automatic reasoning on the propagation of spatial uncertainties becomes possible.

Primitive level

At the primitive level the scope is limited to one primitive at a time. The goal is to find an optimal combination of actuators, sensors and control algorithms to execute the task at hand. Primitive level actions are tightly coupled to the physical devices that will execute the actions. Detailed decisions are made about the geometry of objects.

The core operations at the primitive level are FEED, GRASP, MOVE and MOUNT. In a basic (pick-and-place) assembly process they form the minimum set of operations that can transfer parts from a "loose" state to an "assembled" state. In most assembly systems however, there will be other primitives available.

Actions that contribute only indirectly to primitive level goals -because they only serve to assure the right preconditions, like sensors and gripper change operations, are regarded as overhead. Detailed planning of these actions is done at the primitive level, but they will be included by the part level Planner only when necessary.

2.4 Related research in assembly planning

The main goal of assembly process planning is to generate an optimal process plan, that selects and defines in detail the processes that have to be performed to transform a set of loose parts into a set of assembled products. This plan includes a specification of tools, strategies to use these tools and timing constraints. The quality of the plan can (among others) measured in terms of cycle time and plan robustness.

In section 2.2 analogies were drawn with process planning in parts manufacturing, planning as application of AI-techniques and developments in robot programming. Of course there also have been research projects aiming for the same goal as the project described here: Creating a CAPP system for assembly.

Early work in AI created a very abstract assembly environment called the "Blocks World". Many blocks world planning systems never controlled a real robot. Planning was abstracted to finding a sequence of actions that would create a goal state (some arbitrary complex stack of blocks) from a given initial state (another arbitrary stack) through a series of actions involving stable subassemblies. The world and the family of stacks became more complex as planning algorithms evolved. Important systems were STRIPS [BF81, Vol.2, Ch. D5], ABSTRIPS [BF81, Vol.2, Ch. D6], and BUILD [Fah74]. Techniques that they used include hierarchical problem decomposition, back-tracking, heuristic search and constraint propagation. Some of the algorithms used in product and part level planning were hereby inspired (see chapter 5 and 6).

A second group of planning activities came from robot programming research, where robot languages were extended with features to describe part geometry. Examples of these systems are LAMA, AUTOPASS and RAPT. The latter language specifically focuses on describing geometric relations between features of parts in an APT like manner (see chapter 6).

There is much variation in the amount of research that is being done on the various planning levels. Slightly above the batch level that was defined in the previous section, there is a long history of Operations Research studies on assembly line balancing. Assembly sequence planning has received little attention so far. A few approaches are known from groups with either Artificial Intelligence (AI) or Mechanical Engineering (ME) backgrounds. The part level problem of accumulating spatial uncertainties is mainly addressed by AI groups. Often their context is not assembly but mobile robots.

Research at the primitive level is scattered over many (sub)topics. Much work has been done in the area of (coarse) motion planning. Potential reasons are that motion planning is relatively easy to model abstractly, that the algorithms have a wider application area and that the goal can usually be expressed in simple terms (find the shortest path without collisions). Researchers in motion planning come from many different disciplines. A substantial amount of (mostly theoretical) work has been done on grasp planning and the peg-in-hole insertion problem. In Germany some interesting work has been focussed on specific assembly technologies as screwing and flexible parts insertion.

Most recent work has focussed on more restricted problems, e.g. generating collision free robot motions, or stable grasps.

There is also an example of the interactive approach: It is followed by the TNO-Metaalinstituut in the ESPRIT 384 project [ESP86], [ESP88]. The emphasis of this project is on the user interaction and on modelling the product, the assembly system and the assembly process in a knowledge based representation. A human planner takes all planning decisions, supported by an interactive system that records planner created sequence alternatives. Device assignment is supported by the knowledge base.

Goal of the work as described in this dissertation, is to provide a framework for a generative assembly process planning system. Compared with the TNO-approach, the emphasis is more on the decision making. A top-down approach is followed, starting from the four level reference model, zooming in on more detailed planning activities at each abstraction level. The following section describes the environment and the top level structure of this Planner.

2.5 Global Structure of the Planner

Figure 2.4 presents an overview of the Planner and its environment. The diagram was generated with the software package PROMOD on an IBM PC/AT. PROMOD was the prime tool used for software engineering during the analysis and functional design of the Planner (see section 8.3).

An important input is the description of the product and its components. Core of this product model is a geometric description of all parts, combined with information on the relations between them.

A second important input is a description of the assembly system and its components. As described in the previous chapter, the scope of the work described in this thesis was limited to one workstation. This workstation contains a robot arm, a set of tools, part feeding and transport equipment and may include equipment for special processes. Various types of sensor systems may be available.

The third input concerns the process model. It is the source of process knowledge that is used to make decisions at each level of the four level hierarchy. The process model combines product and system bound information like insertion forces, and obstructed sides of parts in a feeder etc.

For additional input and for support in difficult reasoning processes, an interaction with a human planner is foreseen. In the prototype system, described in chapter 8, the user interface has received relatively little attention. In the DIAC project the user interface is treated as a separate subject (or rather, as two separate subjects, a natural language based interface, and a graphics oriented interface [Jon88, ch. 2].

The output is an assembly process plan, containing a detailed definition of all assembly operations, including a specification of the tools to execute those operations (e.g. robots, grippers), and strategies for using the tools (e.g. grasp strategies, motion trajectories), timing constraints etc. etc.

The two factors that are considered here most important for the quality of the plan are assembly cycle time and plan robustness.

The project focussed on intelligent off-line planning. To demonstrate the ability of the Planner and the quality of the plans that are generated, plans for various products had to be executed in different assembly environments. A simple scheduling mechanism should be sufficient. Scheduling was decided to be a complete topic in its own right and was made a separate project

Figure 2.4: Context of the Assembly Process Planner

within DIAC [Jon88].

Figure 2.5 presents the hierarchical decomposition of the Planner corresponding to the four abstraction levels of the reference model. At the lowest level, the Planner contains two separate functions: `Select_Primitive` and `Simulate_Primitive`. The difference is explained in chapters 6 and 7.

In the development of the prototype Planner, the following restrictions were made:

- At the batch level, the complexity of the process planning problem was restricted by allowing a maximum of one batch (of identical products) in the system (single model batch operation). The batch size was not restricted.
- At the product level, only single element assembly is considered. The prototype sequence Planner was designed to handle products of average complexity.
- At the part level, the Planner works only with solid parts. Springs and other flexible parts can however be handled, either by treating them as solids or by using predefined special handling strategies.
- At the primitive level, the current design is limited to insertion processes whose main assembly motion is a straight line (surface mount, peg-in-hole, screw, snap-fit).

Figure 2.5: Decomposition the Assembly Process Planner

Chapter 3

Batch Level

The batch level is the highest level of the assembly planning reference model presented in chapter 2 (see fig. 2.1). The batch level deals with assembly strategies and optimization of actions for a series of products (one batch). Within the scope of this thesis, the amount of batch level planning is limited, because the work was restricted to the assembly process in a single workstation and a single batch of products (one product type). Within this scope, careful structuring of assembly programs at the lower levels was expected to cover most problems, and development of a batch level planning algorithms was not given high priority. This chapter further deals with the influence of strategical decisions made during system design on the quality of assembly plans.

3.1 Plan optimization

One of the main factors of optimization at the batch level is assembly time. To be more precise: the optimization factor considered in this thesis is *assembly cycle time*, which can be computed by dividing the duration of a certain period by the number of products assembled in that period. It should be noted that the assembly cycle time is not the same as the time it takes to complete one specific product, because several products can be "in process" at the same time.

As described in the first chapter, it is assumed that the product has been designed for assembly and that a flexible assembly system is available. During

assembly process planning the consequences of the decisions made during product and system design become clear, because now actions that will put the product together are assigned to specific devices in the assembly system. Design decisions influence planning decisions at all abstraction levels. Product design e.g. influences the "graspability" of a part and the stability of subassemblies, while the system design e.g. influences robot motion time by constraining the distance from feeder to assembly fixture or making a faster robot available.

The work described in this thesis has been focused on the assembly process in a single (cell structured) workstation. All parts are handled by a single robot and some other tools. Three approaches to cycle time reduction can be distinguished:

- Reduction of "in the loop" time, e.g. by selecting a high speed robot or other tools, or by selecting a "fast" sequence of actions.
- Reduction of overhead time, e.g. by selecting an assembly sequence that minimizes the number of tool changes.
- Concurrent execution of actions, e.g. starting to feed a new part on a bowl feeder directly after one was picked up.

Optimization of the assembly cycle time takes place at all abstraction levels of the reference model. At the lowest (primitive) level the technical applicability of the resources is evaluated. Resource selection is based on "local" knowledge and only the first strategy (select a fast device) can be applied. The higher levels further check the applicability of resources, e.g. on the basis of accuracies. Optimization at these levels is primarily based on the latter two approaches (reduction of overhead and making actions concurrent).

Batch level planning concentrates on decisions for a whole batch. The strategic nature of these decisions makes that batch level planning is closely related to the system layout design. Three areas are considered in this chapter:

1. Parts feeding
2. Tool exchange
3. Transport of parts and products in and around the workstation

The following sections provide more detail.

3.2 Feeding strategies

The first area where important decisions are made during system design of which the consequences become clear during process planning, is the way parts are fed into the assembly system. One can look at parts feeding from different viewpoints. Figure 3.1 gives an overview of part flows between manufacturing and assembly, focussed on the determination of part position. The following categories are distinguished:

Figure 3.1: *Determination of part position between manufacturing and assembly*

1. **Bulk feeding with mechanical orientation.** Parts are brought into the assembly system in bulk. A part specific (mechanical) device (e.g. a bowl feeder) separates and orientates the parts. A delivery track

delivers all parts to a single feed position, which is an advantage if the positions in the assembly system are taught by hand. The feeding device represents a significant investment.

2. **Measuring the position with vision.** Parts are fed in an arbitrary position. The part position is not altered by mechanical means, but measured by a vision system. The assembly robot moves directly to the measured location.
3. **Ordering outside the assembly area.** To save time or forced by a limited space in the assembly area, parts that come in bulk can be ordered automatically or manually outside the assembly area.
4. **Maintain order from manufacturing.** Usually, during certain steps in manufacturing process, parts have a known position. From a global perspective, it may be cheaper to maintain this orientation e.g by stacking the parts on a pallet instead of dropping them in a big box.
5. **Partial on-line manufacturing.** Some parts are small and vulnerable when considered one by one, but relatively stable during manufacturing. Sometimes it is possible to postpone the last manufacturing step (separation) until directly before assembly. Typical examples are parts made from thin metal strip.
6. **Complete on-line manufacturing.** Some parts have such feeding problems (e.g. severe tangling) that direct on-line manufacturing is a solution. A typical example is the on-line manufacture of small springs.

When parts feeding is considered from a process planning point of view, alternatives 3 and 4 are equal: It does not matter where outside the assembly area part ordering takes place. On the other hand, it does make a difference in what form the ordered parts are brought into the assembly area. Many types of magazines or stacks can be used. The largest difference however, exists between single part type magazines and part kits. A part kit contains all parts that are needed to assemble one or a few complete products.

Part kits may seem attractive, because they require little space in the assembly area. The number of different products that can be assembled is not longer restricted by the number of feeders that can be placed in the assembly area. On the other hand, an important part the feeding problem is relocated to the area where the kits are filled. A typical example of a

part kit strategy can be found in the electronics industry. First, resistors, capacitors etc. are glued onto a tape in the right sequence by a machine called the sequencer. Later, the tape (the kit) is mounted on a component placement machine, that places the parts on a printed circuit board.

Parts can be considered as solid bodies in a three dimensional space. Their position and orientation can be described with six independent parameters (an object in 3D space has six degrees of freedom). In current practice vision systems are in use to locate objects on a flat surface (flat board, transport belt). Assuming that the parts are in one, stable orientation (the right side up), only three position parameters have to be measured.

Table 3.1 compares parts feeding methods from an assembly process planning point of view.

Characteristic	Bulk feeding (bowl)	Semi organized (vision)	Magazines and Stacks	Part Kits	On-line Manufacturing
Fixed feeding position	+	-	o	o	+
Problem relocation	+	+	o ¹⁾	-	+
Part redundancy	+	+	+	-	+
Product specific	-	+	o ²⁾	- ²⁾	-
Versatility	-	+	+ ³⁾	+ ³⁾	-
Concurrent availability of multiple parts	-	o	+	+	-
Space requirements in the assembly area	-	o	+	+	-
<div> <div>+</div> positive <div>o</div> neutral <div>-</div> negative </div> <div> <div>¹⁾</div> Potential link to parts manufacturing <div>²⁾</div> Potential cost reduction through general purpose pallets with cheap product specific parts <div>³⁾</div> Important logistic consequences </div>					

Table 3.1: Comparison of parts feeding methods

The following developments are expected:

- Bulk feeding will gradually loose ground to semi-organized (vision-based) and organized (pallet- and kit-based) feeding. The advantage of a single fixed feed position will loose its significance with the development of true off-line programming of positions (directly computed from a CAD model).

- The use of (2D) vision systems for flat board part picking is expected to increase. The vision part of vision guided bin picking in 6 DOF (true 3D vision) could become economically feasible within the next decade. But as bin picking also puts extra demands on the agility of robot and gripper, practical application will be limited.
- The application of part kit feeding will remain limited to specific areas.
- On-line manufacturing will remain restricted to specific part types

Correspondingly, in the experiments with the prototype assembly planner (described in chapter 8), the emphasis was on pallet, kit and vision-based feeding.

3.3 Tool change strategies

Minimizing overhead like tool changes is important if one wants to optimize the cycle time for a given assembly system. Although a significant amount of time can be gained by choosing the right assembly sequence (see chapter 5), important decisions are already taken in the system design phase where a choice for a certain tool exchange strategy is made.

The following strategies are distinguished (summarized in table 3.2):

Fixed gripper A fixed gripper implies that only one type of part is assembled. This is a situation typically found in the large, "Japanese style" assembly lines. One robot assembles one part and there are e.g. 50 robots on one line. Production rates approach mass assembly.

Multi-functional gripper A multi-functional gripper is a gripper with fingers that are designed in such a way, that 2-5 different part types can be handled. The gripper itself can not be exchanged. The situation is typical for a small assembly cell, specifically engineered for one type of product.

Multigripper A multigripper is a turret system on which typically 3-8 small grippers are mounted. Through an index motion, grippers can be quickly exchanged. The multigripper is typically applied for small parts assembly, because the bulky structure of the multigripper constraints the reach of the robot and limit the net allowed part weight.

Finger exchange system A finger exchange system is a relatively cheap alternative for a complete tool exchange system (see below). Many different types of parts can be accommodated, although its application is limited by the gripper base that is used (limiting stroke and maximum gripping force). Usually a gripper with parallel finger motion is applied. The exchange time is relatively long.

Tool exchange system A tool exchange system basically consists of two parts: an active part that is mounted on the robot, and a passive part, mounted on each tool. The interface is standardized and can accommodate electrical, pneumatic and hydraulic connections. Many different grippers can be mounted, but also it is possible to mount other tools on the arm. The disadvantage is that exchange time is relatively long.

Universal gripper Some AI groups are developing a "universal" gripper, consisting of a three or four fingers in a human hand like configuration with three or four DOF per finger. Successful demonstrations of the UTAH/MIT hand at MIT among others showed that many of the basic control problems have been solved. An advantage would be that the need for exchanging grippers would be reduced.

The application of the fixed gripper concept will remain limited to the Japanese style, large series production assembly lines. Process planning for such a system is still interesting because of the large number of stations that has to be programmed, but falls outside the scope of this thesis.

Multi-functional grippers will also be applied in environments where a tool exchange system is available, when the difference between parts is minimal. Multi-grippers will continue to occupy a niche in the market of small product assembly. The addition of a standard tool exchange interface may increase the longer term flexibility (adaptability, planning flexibility) of a multigripper based system.

Tool exchange systems will become more and more standard equipment for every assembly robot, because they also accommodate other tool and their application increases the long term flexibility of the system. Practical application of the "universal hand" in industrial assembly is not expected in the next decade.

In the experiments with the prototype assembly planner (described in chapter 8), the application of the multi-functional gripper, multigripper and tool

Exchange Strategy	Part Types	Exchange time	Remarks
Fixed gripper	1	-	Typical for "fixed" line
Multifunctional gripper	2-5	-	Typical for "fixed" cell
Multigripper	3-8	Short	Limits robot reach Suited for small parts
Finger exchange system	Many	Long	Relatively cheap
Tool exchange system	Many	Long	Accommodates other types of effector
Universal gripper	Many	-	In research stage

Table 3.2: Comparison of tool change strategies

exchange strategies will be evaluated.

3.4 Transport strategies

Overall optimization of batch assembly in a single workstation includes some decisions on the flow of parts and products around the assembly working area. For the incoming flow of parts, several feeding strategies have been described in section 3.2.

Traditionally there have been two concepts for part flow in an assembly workstation (see figure 3.2), one for workstations with a cell structure and one for workstations that are part of an assembly line. In a cell structured workstation typically 5 to 10 part flows converge into the workstation. Parts are fed by the traditional bowl feeder or in some form of magazine. The number of part types is limited by the amount of space that is available around the central element of the assembly cell: the assembly robot.

In the workstation in a line structured system typically a few parts are added to a subassembly that is mounted in a fixture that "flows" through the workstation on a pallet based transport system. Parts are fed by dedicated

Figure 3.2: Part and product flow according traditional concepts

bowl feeders or in some form of magazine.

A promising new concept is the assembly workstation that supports a sub-batch strategy. According to this concept, all part feeding is based on pallets, carrying many parts of a single type. There are two material flow loops through the workstation. One loop carries parts, the other loop carries products "in process". Fig. 3.3 shows the pallet flow in a sub-batch oriented workstation. Products are assembled in small series, the so-called subbatches. The products that circulate in the assembly fixture loop successively receive their parts.

An example: Assume a product of type A , consisting of parts X, Y and Z . First the fixture for product A_1 receives a part X . While the assembly robot is getting a new part X , the fixture loop indexes to a second product A_2 . After this product also received its part X , the fixture loop again indexes to a third product A_3 and so on until all products in the loop have received their first part. While the robot is changing its gripper to one that can handle type Y parts, the part loop indexes to a new pallet with type Y parts. The assembly cycle starts again with the successive insertion of a part Y on product A_1 , product A_2 etc. Finally, all products receive a part Z from a new pallet carrying Z type parts.

Characteristics of a workstation that supports the sub-batch strategy are:

- The distance between the feed position of a part and its assembly

Figure 3.3: *Pallet flow in a workstation that supports a sub-batch strategy*

position is always small. This will result in a short cycle time.

- There is a reduced number of gripper exchanges compared to workstation that exchange their tools for every single product.
- Pallet feeding makes it possible to handle a large number of different part types with one robot, where most other feeding concepts are limited by the amount of space around the robot.

3.4.1 The IVF Mark II System

The sub-batch strategy was developed by the Institute for Production Research (IVF) in Stockholm, Sweden [AG87], [AG88].

Figure 3.4: Layout of the IVF Mark II system

In their Mark II prototype implementation (see fig. 3.4) IVF demonstrates more than just the sub-batch strategy. Their "part" pallet transport system also carries set-up pallets, carrying tools and fixtures for new product

types. The parts pallets are sequenced into pallet trains. The product loop extends outside the robot working area, such that complementary devices (e.g. press, screwdriver) can be included in the assembly cycle. After a sub-batch is completely assembled, the robot transfers the finished products back to special product pallets that come over the "part" transport system.

Flat board feeding plus vision provides high adaptability (new products from a different family) and planning flexibility (new products unforeseen). The same boards can be used for part storage in manufacturing, avoiding double handling of parts. The only constraint is that parts should be placed "right-side-up". Stacking of boards provides high density storage.

3.4.2 An implementation for manual assembly

The main idea behind the sub-batch strategy is the combination of part and product loops. This idea is not new and it does not imply automatic assembly. An example of the sub-batch strategy applied to manual assembly is the implementation of ASCO Controls (see fig. 3.5). On this system several families of industrial electromagnetic fluid valves are assembled.

Parts are fed from trays or low boxes, sorted by part type. Trays and boxes are fed by a paternoster system. The products are assembled on a rotary table. The actual assembly process is executed by hand, but the sequence of actions is familiar: First all products one by one receive their first part, indexing the rotary table after every insertion. When the rotary table has completed its cycle, the paternoster is indexed to a next tray containing a new type of parts. The operator controls the indexing motions of the table and paternoster by foot switches.

Around the rotary table opposite the operator there is room for product specific tooling. Asco e.g. installed a screwdriver unit in this area. The product specific fixtures on the rotary table can be exchanged in a few minutes time.

3.5 The Batch level planner

Within the scope of this thesis, the amount of planning that has to be done by the batch level planner is limited, because the work only considered the assembly process in a single workstation and a single batch of products (one

Figure 3.5: *Layout of the ASCO manual assembly workstation*

product type).

When the workstation operates in the traditional cell or line environment, assembly of a batch often means a repetition of single product assembly programs. At the batch level, there are still some strategical decisions to be made considering parts feeding and tool changing strategies. As argued before, these decisions are strongly related to decisions made during system layout design and therefore not further considered as optimization parameters during process planning.

Sub-batch assembly is a little more complicated, because transport activities of two conveyor loops are interwoven with assembly actions and instead of one, there are now several products concurrently "in process".

Careful structuring of assembly programs at the lower levels was expected to cover most problems, and development of a batch level planner was not given high priority. In the experiments (see chapter 8 some implications of sub-batch assembly are covered.

Chapter 4

Product Level: Plan representation

The product level is the second level of the assembly planning reference model presented in chapter 2 (see fig. 2.1). The product level planner mainly deals with "virtual" assembly actions on parts. These actions have to be sequenced in such a way, that the desired result (assembled products) can be obtained with a short cycle time and with a maximal chance of successful completion.

4.1 Assembly sequence planning: A combinatorial explosion

Generating assembly sequences is a combinatorial process, and the complexity of the problem increases exponentially with the total number of parts involved.

For a simple example product as depicted in fig. 4.1, the problem looks quite simple. Intuitively, we will find several reasonable sequences. We will start with box *A*, add the bottom layer (parts *B* and *C*), and conclude with parts *D* and *E*. Thus we find sequences *ABCDE*, *ACBDE*, *ABCED* and *ACBED*. Some closer observation shows that the sequence *ACEBD* is also attractive.

A sequence planning strategy of generating and testing sequences looks fea-

Figure 4.1: *Example product with 5 parts*

sible. We would start with either one of the five parts, add one of the remaining four etc., until we have a complete sequence, and check this for feasibility. In this way we would have to check $5! (=120)$ sequences.

This approach quickly becomes impossible when the number of parts grows. Eg. for a product with 10 parts we would have to examine $10!$ (over three million) different sequences.

In practical systems the total number of parts ranges from two to more than a hundred (see chapter 1). The "average" assembly system with a cell structure assembles 10-20 parts in total. To be useful in practical applications, a sequence planning mechanism should be able to deal with 20 part products. Therefore, some method for reducing the combinatorial complexity of assembly sequences has to be devised.

More detailed treatment of the actual planning process at the product level is presented in chapter 5. This chapter will briefly introduce assembly sequence planning and then concentrate on the representation of assembly sequence plans.

First, the requirements for such a representation are discussed. After that, some existing methods and their restrictions are described. A new, layered representation will be introduced that provides the required flexibility without growing out to become unmanageably large for a realistic product.

4.2 Representing Sequence Plans

In this section the requirements for a flexible representation of sequence plans are discussed. The sequence representation is intended to be used as an interface between planning and scheduling in the area of flexible assembly. The sequence representation gains in value, when its structure would also support reasoning processes *within* the planning and scheduling modules.

The planner guarantees that the sequence options contained in the representation are technically feasible, the scheduler selects the best sequence according to actual resource availability. When detailed reasoning on resource applicability and allocation is complicated and takes too much time, a major part of the optimization can be done off-line. One could either regard this as very detailed planning or as off-line scheduling.

First some general requirements are treated, followed by a discussion of planning and scheduling aspects. The requirements will be used to evaluate existing methods for plan representation in the next section.

4.2.1 General Requirements

The ideal representation has to describe a set of assembly sequences and should be complete, compact, understandable and flexible:

Complete To be able to optimize the sequence plan it is not enough to represent just one assembly sequence. The ideal representation allows one to describe *any* (theoretically possible) *combination* of sequences. Extreme cases to be considered are the set consisting of only *one* sequence, and the set of *all* theoretically possible sequences.

Compact A sequence representation can be given many forms. One potential dilemma is the choice between (explicitly) listing sequence options versus listing only the sequence constraints. A complex explicit representation is often associated with increased memory requirements and long computation times. On the other hand, a more implicit and compact form usually leaves much work to be done later.

The ideal representation provides a compact balance between implicit and explicit forms. The structure of this representation is such, that its size is proportional to the complexity of the sequence plan encoded, not to the number of alternatives within that plan.

Understandable The representation will be used in a flexible assembly environment. Changes in product structure and system layout will occur frequently. The assembly system is expected to be able to cope with a part of these changes autonomously, but in some cases human intervention will be required. A sequence representation with a graphical base or a graphical equivalent could help human understanding.

Flexible The representation would be more valuable if it did not prescribe the exact amount of detail in each sequence plan. Some flexibility would e.g. allow complex products to be dealt with in two stages, first a global analysis, then with more detail. Also, this flexibility would provide an easy growth path to more intelligent assembly systems, that would do part of the process planning on-line and use a more abstract input.

4.2.2 Assembly Sequence Planning Aspects

The assembly process at the product level (the level that deals with assembly sequences) can be modelled as a sequence of discrete steps, where each step represents the assembly of one part or of a group of parts. Each intermediate state describes the product in terms of parts that have been assembled so far. In the *Start* state all parts are loose, in the *Goal* state all parts are assembled. Transitions between states correspond to process steps (assembly actions) that add one or more parts.

Planning of assembly sequences can be described as a search problem in the space of assembly states. The resulting plan is a set of feasible paths between the start and goal state. First it has to be established which of the theoretically possible states are allowed and reachable. After that, this set of states can be searched for feasible paths from the start state to the goal state.

To be applicable during the planning phase, the plan representation has to include an explicit description of both states and transitions. These elements are important to be able to incorporate the effects of accessibility and stability on the assembly sequence:

Accessibility influences part sequences by blocking certain transitions from one state to a successor state. An example is illustrated in figure 4.2-a. Part *C* can not be inserted because it is blocked by *D*. To support such accessibility checks, the sequence representation should contain explicit references

to transitions.

Figure 4.2: *The influence of accessibility and stability*

Stability of a subassembly under given circumstances (gravity, disturbing forces) determines whether that subassembly is allowed as an intermediate state. E.g. under normal circumstances (gravity, no special tools), the situation as depicted in fig. 4.2-b is unstable, and should not be allowed as an intermediate state. Since the stability of a subassembly strongly depends on part geometry and subassembly *composition*, such information should be available in or through the sequence representation.

For the purpose of plan optimization the plan representation should allow an explicit description of alternative sequences. One state can be the result of several actions, but also be the starting point for several new actions. Thus, in the sequence representation a state should be allowed to have multiple predecessor states and multiple successor states.

The selection of the optimal assembly sequence is also influenced by resource specific overhead like the time needed for tool changes. This overhead not only depends on the current part to be handled, but is influenced by the total situation (which other parts have been assembled), and recent history (which part has been assembled last). Thus, the ideal plan representation would not carry an explicit reference to only one subassembly per state, but to a complete description of the whole situation in terms of all parts in the product.

4.2.3 Scheduling

The Scheduler expects a sequence plan that leaves some sequence options open, giving the scheduler an opportunity to produce better schedules. As mentioned before, part of the resource selection and allocation can be done off-line, during sequence planning or as an off-line scheduling activity. Some examples:

- Improve the system performance by splitting up a large assembly task into subtasks and assigning these to multiple machines (concurrent assembly of subassemblies), or by grouping common subassemblies in larger batches (economy of scale).
- Reduce the adverse effects of uncertainty in parts availability (opportunistic scheduling, see [Fox87])
- React to changes in the availability of system resources (exploiting functional redundancy of system components to reduce system down-time)

Admissible transitions

The Scheduler takes into account both the constraints imposed by the planner (the feasible sequences) and the constraints imposed by the execution time environment. An operation necessary to achieve a new state of the world as specified by a plan can only be performed after all required predecessor operations have been performed and all the necessary resources are available. An adequate sequence representation would make it easier for the Scheduler to keep track of the current state, and provide information of available (admissible) next process steps. The introduction of a cost factor associated with transitions (of which assembly time would be an important element) would simplify the selection of a cheap path. Because the planner takes partial care of resource allocation and sequence optimization, such a cost factor is of importance there too.

Multiple successor states

There are several causes for uncertainty in the availability of parts. A distinction can be made between uncertainty about the identity of the parts, uncertainty about their position and orientation, uncertainty about their presence, and uncertainty about their quality.

The assembly process takes place in a relatively structured environment. The parts are known and details about their size and shape should be available, so there is no need to worry about recognizing unknown parts. Parts that do not look familiar can simply be rejected. As products will be assembled in larger quantities, replacement parts should be available. However, part rejection could mean some delay in part availability (presence uncertainty).

It is hard to find a reason for feeding parts at random in a jumbled heap or mixed in a bin. In general, it should not be difficult to keep parts separate according to part type from the moment they are produced. Position uncertainties can be resolved either by mechanical means or by position sensors, which can again imply some presence uncertainty.

Productivity loss by part presence uncertainty can be reduced by allowing alternative sequences: The probability of progress is increased by enlarging the number of admissible successor states.

Minimize on-line work

To be able to react quickly to changes in the system's environment the Scheduler should run on-line with a cycle time that is of the same order of magnitude as the system's cycle of plan execution. As is argued in the previous chapter, it is not likely that schedules will be made for every single product. The smallest time-span will cover a (small) group, a batch or sub-batch of identical products. Still, for acceptable response times, it will be necessary to minimize the amount of on-line work in the scheduler. The following measures can be taken:

- Make sure that all technically impossible sequences are eliminated from the plan during planning, so that no plan checking has to be done during scheduling. In complex cases, one could go further and have the planner remove some more unlikely sequences from the plan.
- List all sequence *choices* in the plan explicitly (instead of listing only sequence *constraints*)

4.3 Candidate methods

Several new methods to describe assembly sequences have been reported. Most of them acknowledge the limitation of currently used precedence graphs,

prescribing a fixed part ordering. New approaches include AND/OR graphs [HdMS86a], constraints [FK85], constraint networks [Fox86] and Petri nets [MA86]. Other well-known approaches are the triangle table and temporal constraints. In the following subsection these methods will be discussed. The five part product of fig. 4.1 will be used in the examples.

4.3.1 Triangle Table

				E
			D	
		C	<	<
	B		<	
A	<	<	< *	< *

Figure 4.3: Triangle table for the 5 part product.

The triangle table is a convenient way to represent binary relations between objects in more or less graphical form. In fact it is just another form of $n * n$ matrix, where N is the number of parts in the product, and where the choice of sequence constraints renders one half of the table redundant.

Fig. 4.3 presents an example. The $<$ indicates, that the element at the left has to be assembled before the element on top of the column. In this special case there are no $>$ relations, that would indicate the reverse. The $*$ indicates a redundant constraint, which could be derived from other constraints, e.g. $A < C$ (A before C) combined with $C < D$ (C before D) implies $A < D$ (A before D).

The main advantage of the triangle table is that it is very compact (it only contains $N^2/2 - N/2$ elements). On the other hand, it is impossible to describe some alternatives (e.g. *A, B, C in any order but B not last*) in one table, and an explicit description of possible next steps and intermediate states is not available.

4.3.2 Precedence diagram

Figure 4.4: Three different precedence diagrams for the 5 part product.

Precedence diagrams are often used in industry. There are several conventions. According to one convention (depicted in fig. 4.4-a) the diagram represents a strict sequence of actions. This diagram is read top-down and left to right. At the top left, parts *A* and *B* are joined into subassembly *AB*. In the next step this subassembly is combined with part *C* to form subassembly *ABC* etc. The product thus grows from the top line, where all parts are loose, to a complete product at the bottom line.

According to another convention, the precedence diagram can also describe subassemblies other than straight-line descendants from the first part (see fig. 4.4-b). Here parts *D* and *E* are first joined into subassembly *DE*, which later is combined with subassembly *ABC* to complete the product.

According to a third convention, the precedence diagram only describes direct precedence constraints between parts (an example is depicted in fig. 4.4-c).

All three versions of the precedence diagram are compact. In the first two, the composition of subassemblies is explicitly represented. Sequence options are only partially explicit. It is always clear when some part has to be assembled before further assembly can take place. The first convention allows only a linear sequence of assembly. The other conventions do not clearly indicate any precedence relation between subassemblies. E.g. in (fig. 4.4-b) it is not clear whether *AB* or *DE* should be assembled first.

4.3.3 AND/OR Graph

The AND/OR Graph can be regarded as a concatenated set of precedence graphs [HdMS86a], [HdMS86b]. Each node in the graph represents a group of parts (a "subassembly"). The links between the nodes are formed by hyperarcs, which indicate how several "children" can be combined (assembled) into one "parent". A hyperarc thus represents an *AND* clause over all "child" nodes. Alternative ways to assemble a "parent" are indicated by alternative hyperarcs descending from it, thus establishing the *OR* part of the graph.

The AND/OR graph explicitly represents the composition of subassemblies in its nodes. These nodes can have multiple successors, so many alternative sequences can be represented explicitly in one graph. For each product there is only one *full AND/OR graph*, a theoretical structure that includes every possible subset of parts as a "subassembly". The levels are numbered from 1 to N , where level 1, the lowest level, contains N nodes (all loose parts) and level N contains one node (the end product with N parts). level l contains $\binom{N}{l}$ nodes. The total number of nodes in the full graph is:

$$n_f = \sum_{l=1}^N \binom{N}{l} = 2^N - 1 \quad (4.1)$$

Stability and accessibility constraints limit the number of *allowed* nodes, and reduce the number of nodes and hyperarcs in the graph. For the five part product that is used here as an example, the full graph would have $2^5 - 1 = 31$ nodes. After elimination of unstable subassemblies and subassemblies involved in inaccessible transitions only subassemblies *AB*, *AC*, *ABC*, *ACE*, *ABCD* and *ABCE* remain. Together with the loose parts and the end product, this means 12 nodes.

Fig. 4.5 presents an example of an AND/OR graph for the 5 part product, after elimination of inaccessible and unstable states and transitions. The figure demonstrates the compactness of the AND/OR graph compared with a set of precedence graphs: With the addition of just one node (*DE*) and two connecting hyperarcs, the graph can represent several new sequences (involving node *ABC*). Note that under usual circumstances, node *DE* will represent an unstable subassembly. The capability of representing such situations is important to the Planner.

In the example only binary hyperarcs are considered (one parent has exactly

Figure 4.5: AND/OR graph for the 5 part product.

two children), corresponding to the assembly of one part or subassembly at the time. The AND/OR graph representation could however easily be extended (e.g. ternary hyperarcs could be used to indicate concurrent assembly of three parts).

Fig. 4.5 illustrates that in AND/OR graphs precedence relations between parts and subassemblies are not always clear, e.g. the hyperarc combining *ABC* and *DE* does not indicate any precedence relation between the two subassemblies represented by both nodes.

Another problem is the size of the representation. Although the AND/OR graph is much more compact than some explicit enumeration of sequences, its size still grows exponentially with the number of parts. A good indication for the complexity of an AND/OR graph is the number of nodes in a graph. For a product with N parts the maximum number of nodes in the graph is $2^N - 1$, corresponding to the number of non-empty subsets of a set of N elements. For a product with 20 parts that would be roughly 1 million nodes.

4.3.4 Temporal Logic

Temporal logic presents a different approach to representing assembly sequences. The method focuses on actions that take place in certain intervals. All actions have either an explicitly or implicitly specified duration and concurrency of actions is implicit unless specified otherwise.

Based on a formalized description of relations between intervals (interval algebra [All83]), temporal logic provides a much richer way of expressing sequence relations than the representations treated so far. In this symbolism, one can e.g. specify that one assembly action has to be completed *before* another action may start, or that an action takes place *during* another action. Table 4.1 presents an overview of primitive temporal relations.

Using temporal logic, an assembly sequence plan can be described as a network of constraints. For the five part product, the plan could e.g. be described as follows: $A(< m)B$, $A(< m)C$, $E(> mi)C$, $D(> mi)B$, where A , B etc. form nodes representing the actions *Assemble A*, *Assemble B* and the arcs between those nodes are labeled with the primitive temporal relations that exist between those actions (see fig. 4.6).

Semantic interpretation of the network is possible through the transitive

Relation	Symbol	Example
X equals Y	$=$
X before Y Y after X	$<$ $>$
X meets Y Y met by X	m mi
X overlaps Y Y overlapped by X	o oi
X starts Y Y started by X	s si
X during Y Y contains X	d di
X finishes Y Y finished by X	f fi

Table 4.1: Overview of primitive temporal relations. The bars X and Y represent time intervals, the horizontal axis corresponds to elapsed time.

properties of the primitive relations. E.g. from $A(< m)B$ and $D(> mi)B$ can be inferred that for any B with a finite duration, $A(<)D$.

Overlap between actions can be represented in the notation. E.g. the specification that the assembly of D may start before assembly of E is complete, translates into $D(o > mi)E$. This could be very useful in situations where several devices can function in parallel.

When temporal constraints are used to describe an assembly sequence plan, incremental adaptation of this plan is possible: The future possibilities are encoded in constraints (restrictions on actions) instead of in states of the world. This gives a scheduler more freedom to react intelligently to changes in the environment.

Disadvantages of temporal logic are that the composition of subassemblies and complete sequences can not be explicitly represented. Furthermore, the need to represent the sequencing problem as a set of relations between actions expressed in temporal logic presents a coding problem that is not straightforward.

Figure 4.6: Network of temporal constraints for the 5 part product.

Solutions (possible sequences) are found by propagating constraints. This is very complex (constraint propagation is a NP-Hard problem) and tools to support constraint propagation and examine sets of constraints (e.g. to check consistency) are not readily available.

4.3.5 Petri Nets

Petri Nets form a potentially attractive representation method for assembly sequences because there is a substantial base of theoretical work available that could be of help to formalize the sequence description. Researchers in Edinburgh are already applying Petri nets in this way [MA86].

Petri nets are a form of state transition networks that correspond to a certain formalism regarding the network structure. They can be used in several ways to represent assembly sequences. One example is shown in fig. 4.7. Here the Petri net is used to describe the assembly process in terms of part states and sequence constraints.

In this Petri net there are two types of nodes, circles and bars. A circle, or *place* in Petri net terminology, represents one state of one part. Each part can be in exactly two states, loose (e.g. $\bar{A}$) or assembled (e.g. A). Which state a part is in, is indicated by a label or marking (the *token*), which is

Figure 4.7: Petri net for the 5 part product

often depicted as a black dot in the active place. The bars (or *transitions*) govern the transition of tokens from one place to another. A transition is enabled when there is at least one token available in all directly preceding places (implying an *AND* constraint). A set of *firing rules* determines which of the transitions may fire next. The state of the product being assembled corresponds to the distribution of tokens over the network. One such distribution is called a *vector*.

In the Petri net of fig. 4.7, some tokens are allowed circulate in loops. This is used to express the assembly sequence constraints. The constraint $A < B$ is e.g. encoded by incorporating a loop from A to A in the transition from $\bar{B}$ to B . The loops make the network difficult to read.

As shown in fig. 4.8, the incorporation of special enabling places (indicated with a $*$) makes the structure of the network more explicit. Following the modern convention, the transition bars have been replaced by transition boxes. The total amount of tokens in this network is not constant; tokens are created and deleted by the transitions when appropriate. The horizontal axis of this diagram roughly corresponds to a time axis.

In the Petri nets of fig. 4.7 and 4.8, subassemblies and sequences are not explicitly represented. This is not an inherent restriction of the Petri net formalism. Subassemblies could for instance be represented as vectors of token distribution; sequences could be made explicit by ordered sets of vectors. For products with many parts however, this would be a rather tedious job.

Another possibility would be to model subassemblies explicitly as places in the Petri net. The result could be like fig. 4.9, a structure that is very similar

Figure 4.8: Petri net for the 5 part product with explicit enabling conditions

to the AND/OR graph. A difference with the original AND/OR graph is the explicit representation of the sequence constraints in the form of box nodes instead of the direct hyperarc connections. Also the Petri net supports the use of tokens to emulate the assembly process.

The representation of fig.4.9 suffers from the same problems as the AND/OR graph: For products of 20 or more parts it will become too large to manage, and sequences are not completely described.

4.3.6 Constraint Networks

Constraint Networks [Fox86],[Fox87] provide an improvement over AND/OR graphs, by representing sequences more explicitly, offering a more compact structure. On the other hand, subassemblies and intermediate states are not explicitly represented.

Figure 4.10 presents a set of constraint networks for the 5 part product. In fact there are two (partial) networks, each representing one conjunctive clause over the set of admissible assembly sequences. The left part of the figure could e.g. be read as: *((A before B) and (A before C) and ((B and C) before D) and (D before E).*

The easiest way to represent the subassemblies would be to "pebble" the

Figure 4.9: Petri net for the 5 part product including subassemblies

Figure 4.10: Set of constraint networks for the 5 part product

nodes that have been visited (like a Petri net token that would leave some mark in a node). This would probably be useful for the scheduler, but the planner would rather have the subassemblies and intermediate states represented more explicitly.

4.3.7 Evaluation

Table 4.2 gives an overview of the features of existing methods for representing assembly sequence plans.

Most of the criteria that were used for the evaluation have been described in the previous section. The understandability requirement has been interpreted as a requirement on the graphical equivalent of the representation: It should describe both states and transitions. One extra criterion is the availability of tools (algorithms) to work with the representation.

The methods described in the previous paragraphs can be globally divided into two groups: A group of constraint based methods (triangle table, temporal logic, constraint networks, and the constraint based variants of the precedence diagram and petri net), and a group of methods capable of rep-

	Triangle Tables	Precedence Diagram	Temporal Logic	AND/OR Graphs	Petri Nets	Constraint Networks
Complete	.	.	•	○	○	•
Compact	•	•	•	○	○	○
Graphical	○	•	○	•	•	○
Flexible	○	○	○	○	○	○
Mult. Succ.	○	○ ¹	•	•	• ¹	○
Expl. Trans	.	•	○	○	• ¹	•
Expl. Sub.	. ²	•	. ²	•	• ¹	. ²
Expl. State	.	• ¹	.	○	○	.
Tools	○	• ¹	.	○	○	○

.	Not or marginally supported
○	Some support available or possible
•	Well or easily supported
¹)	Using the best variant
²)	Possible by introducing abstraction levels
Mult. Succ.	Representation allows multiple successors
Expl. Trans.	Explicit reference to transitions
Expl. Sub.	Explicit reference to subassemblies
Expl. State	Explicit reference to states
Tools	Availability of tools to work with representation

Table 4.2: Comparison of existing sequence plan representations

Figure 4.11: Triangle tables describing the 5 part product at two abstraction levels.

representing subassemblies in a more explicit way. The constraint based representations are very compact, but in their basic form they do not contain enough explicit information.

All methods provide a form of flexibility: they could be extended with abstraction levels. For the methods of the first group this presents an opportunity to create direct references to subassemblies. Fig. 4.11 presents an example for the triangle table. The five part product is assumed to contain a subassembly BC. The composition of this subassembly is described in a separate triangle table F. Fig. 4.12 presents a similar example for the precedence network. The introduction of abstraction levels does not make these methods complete.

In the second group (precedence graphs, AND/OR graphs and Petri nets), the AND/OR graphs present an advantage over ordinary precedence diagrams: They are more compact when it comes to representing sets of sequences. For more explicit reference to transitions, e.g. to keep track of all *feasible transitions* during actual assembly, a Petri net like implementation of the AND/OR graph appears to be preferable. The only representation that explicitly describes all intermediate states is a restricted form of the precedence diagram. None of these methods is complete.

In the next section a new, complete representation method is introduced that does explicitly describe transitions, subassemblies and intermediate states. Abstraction levels are used to keep the representation compact.

Figure 4.12: *Precedence diagrams describing the 5 part product at two abstraction levels.*

4.4 The Layered Assembly State Transition Diagram

The Layered Assembly State Transition Diagram (LASTD) is a new method for representing assembly sequences. The first paragraph of this section describes how the LASTD exploits redundancy in parts. The second paragraph introduces the basic structure of the LASTD, the single level ASD. The last paragraph describes how part sequence groups (clusters) can be exploited to create layers of abstraction that keep the LASTD compact.

4.4.1 Exploiting part redundancy

Parts are the building blocks of every assembly. Analysis of products used in earlier and related work in the Laboratory [Ver85], [Jam88], [Mar88a], showed that that in many products certain parts occur more than once. This finding was supported by the literature review described in chapter 1. Although the "average" cell structured assembly system assembles 10-20 parts in total, these parts come in 6-10 types only. This paragraph shows how this redundancy can be exploited to reduce the complexity of the sequence representation.

First, a distinction has to be made between parts before and after they are assembled in a product. In fig. 4.1 a product was shown, consisting of five

Figure 4.13: Positions and parts of the 5 part product

parts. Now let us assume that there is some redundancy in parts, e.g. that parts *B* and *E* and parts *C* and *D* are of the same type. This is indicated by relabeling parts into *B*₁, *B*₂ and *C*₁, *C*₂. Making an explicit distinction between the parts before they are assembled (indicated with lowercase plus number) and their potential positions in the product (indicated with uppercase plus number) results in 10 unique labels as shown in fig. 4.13.

Figure 4.14: Physically different, but functionally equivalent products

Part *a1* is a box of type *A* and can only be used in position *A1*. Parts *b1* and *b2* are blocks of type *B* and could be used in either position *B1* or *B2*. In the same way, parts *c1* and *c2* are blocks of type *C*, to be assembled in positions *C1* and *C2*. Exchanging parts *b1* and *b2*, *c1* and *c2* will lead to four physically different, but functionally equivalent products. Two of these variants are depicted in fig. 4.14.

In a rationalized assembly process, parts of the same type are fully exchange-

able and identical within small boundaries. Thus making a distinction between products like the two shown in fig. 4.14 is not necessary.

On the other hand, part positions are *not* interchangeable. It should be possible to discriminate between subassemblies that consist of exactly the same parts, but in different positions. An example is shown in fig. 4.15: The two subassemblies $A1B1C1$ and $A1C1B2$ consist of the same parts, and are uniquely identified by the part positions.

Figure 4.15: Two subassemblies of the 5 part product consisting of the same parts

In the rest of this thesis, products will be described by part-position representations.

4.4.2 The ASTD

The basis of the LASTD is the single level Assembly State Transition Diagram or ASTD. An ASTD is an acyclic, directed graph, describing (a part of) the assembly process as a network of states.

The ASTD explicitly indicates which parts are loose, which are assembled, and the composition of subassemblies. It also describes all admissible sequences explicitly. Furthermore, it has a graphical base that is easy to understand.

The ASTD consists of nodes (*states*), that correspond to the composition of the product and its subassemblies, and directed arcs (*transitions*), that correspond to assembly actions. At each node there is a list that completely indicates the status of all parts of the product. The elements in this list

refer to parts, or to (sub)assemblies. In the start state all parts are "loose" (there are no subassemblies). In the final state (the goal state) contains a single assembly (the product).

During each transition, one element (part or subassembly) is assembled (put in a fixture or joined with an already assembled element into one new element). Each path through the ASTD from the start state to the goal state corresponds to one complete assembly sequence.

In the state description, loose parts are indicated with a bar ($\bar{}$). Assembled elements are separated from each other and from the loose parts by slashes. Each state contains a number of elements that corresponds to the number of assembly actions that have been executed. When there are n parts in total, there are n elements to start with (all loose parts). After the first step, there are still n elements, one of which is an assembled part. Each step thereafter reduces the element count by 1: there are $n - 1$ elements after completion of step 2, $n - 2$ elements after step 3 etc.

Figure 4.16 presents an example of an ASTD for the 5 part product. Since there are 5 parts, it takes 5 steps to get from the start state $\overline{ABCDE}$ to the goal state $ABCDE$. The start state contains 5 elements (5 loose parts), the goal state contains 1 element (the product). The intermediate state $ABC/\overline{DE}$ (in the middle of fig 4.16) contains three elements: the subassembly ABC and the loose parts $\overline{D}$ and $\overline{E}$.

The ASTD allows multiple subassemblies to exist in one state. For the 5 part product one could imagine a state $AB/CE/\overline{D}$, consisting of three elements: Subassemblies AB and CE and the loose part $\overline{D}$.

A problem of the ASTD is its size. The number of nodes in the graph depends on the convention chosen, and on the complexity of the product. In the simplest case there is only one fixed assembly sequence so that each level contains only one node, and for a product with N parts there are $N + 1$ nodes in total. On the other hand, when there are only a few sequence restrictions, the size of the ASTD can increase exponentially with the number of parts.

Two methods are used to overcome this problem. In the first place, the number of subassemblies per intermediate state is restricted to one, which corresponds to the situation where one robot works on one product at the time, an assumption that is made throughout the rest of this thesis. In the second place, sequence groups are introduced to create abstraction levels.

When the number of subassemblies per intermediate state is restricted to

Figure 4.16: ASTD for the 5 part product. Note that A, B, C etc. encode part-positions.

one, the ASTD contains at most $N!$ sequences and 2^N nodes. For this case the complexity of the ASTD corresponds to that of the AND/OR graph.

The general structure of the ASTD allows more than one subassembly per level. However, the advantage of the explicit sequence description then quickly turns into a disadvantage, because duplicate subassemblies will occur and the number of states in the ASTD will grow explosively.

An example: for the 5 part product, consider the last step in the assembly sequence. There is a set of "before-last" states that are only one transition away from the complete product. In all these states, the parts can be divided into two elements. When only one subassembly per state is allowed, there are five states in the set, with a similar form as $ABCD/\bar{E}$ (one four part subassembly and one loose part).

When 2 subassemblies per state would be allowed, there are more "before-last" states. The number of ways in which n parts can be divided in exactly k groups, g_{nk} , is given by:

$$g_{nk} = \frac{1}{k!} \sum_{i=0}^k -1^i \binom{k}{i} (k-i)^n \quad (4.2)$$

In this case, $n = 5$ and $k = 2$, and equation 4.2 indicates 15 possibilities, five of the form mentioned before, plus ten of the form ABC/DE (two subassemblies, one with two, the other with 3 parts). For larger products, or going back more steps, the number of states per level will grow even more dramatically. Also, duplicate subassemblies will occur, e.g. the subassembly ABC in state ABC/DE will also occur in earlier states like $ABC/\overline{DE}$. The value of this redundant information is low.

For many practical applications, the representation of one subassembly per intermediate state will be sufficient. Complex products could be divided into subassemblies at an early stage, each of which would be treated as a separate product. In the final assembly, each of these subassemblies would then be treated as a single, although somewhat complicated, part. This strategy is effectively equivalent to the introduction of abstraction levels. The following section will show that this strategy is not restricted to subassemblies consisting of a physically stable groups of parts, but that it can be extended to a more general concept of sequence groups.

4.4.3 Sequence groups

In many assemblies, some (physical) structures can be recognized, that make many of the theoretically possible assembly sequences improbable, irrelevant or just impossible. Sequence groups, or *clusters* as they are called, are meant to focus the attention on the more probable sequences.

A cluster is defined as *a group of parts that belong together logically because they have some assembly features in common, because of which they will normally be assembled as a group*. The combined parts do not necessarily form a stable subassembly. A key assumption is that once the assembly of a group is started, the sequence may not be interrupted by parts from outside that group.

Consider the product of fig. 4.17. The product is a simplified version of the 5

Figure 4.17: Example product with 4 parts

part product used before. The top block (D) is a lid that completely covers parts B and C . The figure can be seen as a schematic view of a gearbox with two gears in it.

In theory, the 4 parts can be assembled in $4! = 24$ sequences. Grouping the two "gear" parts B and C into a cluster E , will reduce the number of sequences to be considered, because sequences that interrupt the assembly of cluster E are ignored. Examples of such ignored sequences are $ABDC$ and $CADB$, sequences that had little practical value anyhow. The total number of sequences left to be considered is halved to 12. For larger part counts, much larger reductions can be achieved (see chapter 5).

Definitions of cluster types and their derivation from the product description will be presented in the next chapter. The last section this chapter will show how clusters are used to create abstraction layers in the LASTD.

4.4.4 A layered structure

The advantageous properties of the LASTD over its single level counterpart will be illustrated by an example. For each number of parts there exists only one "full" or complete ASTD, depicting all possible combinations of all parts. Then, for each product variant there will be many "reduced" diagrams, that show the possibilities after product limitations are introduced.

Figure 4.18 presents the complete (single level) ASTD for the 4 part product (see fig. 4.17). As explained in the previous section, there is only one subassembly per state. In the ASTD, transitions that are inaccessible, ir-

Figure 4.18: ASTD for the 4 part product. Inaccessible transitions and transitions to irrelevant states are marked with X

relevant, or lead to irrelevant subassemblies have been marked with X. All these transitions relate directly or indirectly to the creation of subassembly AD (consisting of box A and lid D), before both other elements are in place.

Transitions $(AD/\overline{BC} \rightarrow ABD/\overline{C})$ and $(AD/\overline{BC} \rightarrow ACD/\overline{B})$, corresponding to the insertion of B and C into AD , are inaccessible. The same holds for transitions $(ABD/\overline{C} \rightarrow ABCD)$ and $(ACD/\overline{B} \rightarrow ABCD)$. As there is no way of creating a finished product from the states $ABD/\overline{C}$ and $ACD/\overline{B}$, these states might be "reachable", but getting there is irrelevant. Thus, other transitions to these states, such as $(AB/\overline{CD} \rightarrow ABD/\overline{C})$ and $(CD/\overline{AB} \rightarrow ACD/\overline{B})$, become irrelevant in turn.

In a similar way, because transitions $(AD/\overline{BC} \rightarrow ABD/\overline{C})$ and $(AD/\overline{BC} \rightarrow ACD/\overline{B})$, have become irrelevant, state $(AD/\overline{BC})$ and its incoming transitions $(A/\overline{BCD} \rightarrow AD/\overline{BC})$ and $(D/\overline{ABC} \rightarrow AD/\overline{BC})$, become irrelevant in turn. Thus, the sequence constraints propagate through the ASTD.

The result of this constraint propagation is shown in fig. 4.19. States AD , ABD and ACD , and their connecting transitions have been deleted.

Now, consider the introduction of cluster E , consisting of parts B and C . The cluster makes it possible to describe the assembly at two abstraction levels. At the top level, cluster E is treated as a single, though somewhat special part. At the second level the composition of E is described. Together they can be represented in the Layered Assembly State Transition Diagram, or LASTD (fig. 4.20).

The LASTD of fig. 4.20 consists of two ASTD's. The ASTD on the left represents the top level. It describes the assembly of three "parts", A , D and E . Because E is not an ordinary part but a cluster, its assembly is indicated with a double arrow. The ASTD on the right represents the assembly of cluster E .

Again, accessibility constraints can be used to reduce the number of sequence alternatives. The inaccessible and irrelevant transitions have been marked with X in fig. 4.20. The only transition that is directly marked as inaccessible is $(AD/\overline{E} \Rightarrow ADE)$. This constraint propagates back to state $(AD/\overline{E})$ and its preceding transitions $(A/\overline{DE} \rightarrow AD/\overline{E})$ and $(D/\overline{AE} \rightarrow AD/\overline{E})$. The reduced LASTD is shown in fig. 4.21.

Comparing figs. 4.18 and 4.20 shows that the LASTD is more compact than the ASTD. The smaller number of crossed out transitions in the LASTD indicates that the LASTD is also a more powerful basis for reasoning on

Figure 4.19: Reduced ASTD for the 4 part product. Inaccessible and irrelevant transitions and irrelevant states have been deleted

a) Product

b) Cluster E

Figure 4.20: LASTD for the 4 part product. Inaccessible transitions and transitions to irrelevant states are marked with X

Figure 4.21: Reduced LASTD for the 4 part product. Inaccessible and irrelevant transitions and irrelevant states have been deleted.

sequences. This advantage partially stems from the introduction of abstraction layers (deleting one double-arrowed cluster transition corresponds to deleting many sequences), partially from the definition of a cluster (non-interrupted sequence).

4.5 The LASTD versus existing methods

	Prec. Diag.	AND/OR Graphs	Petri Nets	LASTD
Complete	.	o	o	•
Compact	•	o	o	•
Graphical	•	•	•	•
Flexible	o	o	o	•
Multiple successors	o ¹	•	• ¹	•
Explicit reference to transitions	•	o	• ¹	•
Explicit reference to subassemblies	•	•	• ¹	•
Explicit reference to states	• ¹	o	o	•
Availability of tools	• ¹	o	o	o
. Not or marginally supported o Some support available or possible • Well or easily supported ¹) Using the best variant				

Table 4.3: Comparison of the LASTD with some other representations

Table 4.3 compares the features of the LASTD with those of several existing methods for representing assembly sequence plans.

The LASTD is complete in the sense that it *can* represent all combinations of sequences. Usually, this feature will not be exploited completely, because the size of the representation grows exponentially with the number of sequences explicitly represented. Abstraction levels are introduced to focus the attention on relevant sequences. The quality of the representation can be expressed in terms of the number of relevant sequences that is or is not included. This quality depends on the heuristics that are used to recognize the sequence group (cluster recognition, see chapter 5).

The LASTD is more compact than the other representations because its abstraction levels make it possible to exclude large sets of irrelevant sequences in an early stage. Although no report was found in literature, abstraction levels could be introduced for the other representations as well (see section 4.3.7). The effect would be similar, the results still depend on the definition of the sequence groups. Precedence diagrams are more compact than the LASTD, but offer less explicit information and can not represent

all alternatives.

The graphical representation of the LASTD describes states (including the composition of subassemblies) and transitions. The layered structure of the diagram corresponds to the physical structure of the product.

The use of part positions instead of referring to individual parts adds to the flexibility of the LASTD. Parts of one type can be assembled in any position suited for that part type. No report on this idea was found in literature, although it is easily introduced in most methods.

The LASTD contains more explicit information per state than any of the other methods. The basic version of the precedence diagram is the only other method that completely describes the state of all parts. The reason is that this diagram only represents a linear assembly sequence with exactly one state (containing one subassembly) per level.

Tools for creating and searching sequence plans are not readily available for any of the representation methods. To create such tools is relatively simple for the basic version of the precedence diagram; very complex it is only for networks of temporal relations. The effort for the other methods (including the LASTD) is moderate and comparable. Data structures and algorithms can be based on general descriptions of graph creation and graph search found in literature. The introduction of abstraction levels does add some extra complexity.

In general, it can be concluded that the LASTD combines several features of existing sequence representation methods with some new ideas that keep it compact and flexible. The next chapter will go deeper into the process of assembly sequence planning, using the LASTD as the main representation in several stages.

Chapter 5

Product Level: Sequence Planning

This chapter describes the design of the product level sequence planner. First the structure of this planner is detailed, followed by a discussion of cluster recognition, LASTD reduction heuristics and expansion of part level actions. The chapter concludes with an example of LASTD generation for a nine part product.

5.1 Structure of the Sequence Planner

5.1.1 The Environment

The sequence planner operates at the second level of the assembly planning reference model presented in chapter 2 (see fig. 2.1). Corresponding to the global project aim of generating plans for assembly systems that are flexible in many aspects, the sequence planner has to support plan generation for a broad spectrum of products, and create sequences in a way that allows plan optimization and intelligent exception handling.

The sequence planner focuses on the assembly sequence for a single product. The available set of resources is assumed to be limited and fixed (globally assigned and optimized by the batch level). Still, the amount of combinations and sequences of resource use will be very large. The sequence planner interacts with part and primitive level planners to obtain detailed informa-

tion on how parts can be grasped, when and how to use sensors etc. The results include a cost estimate for each action.

5.1.2 The Global Structure

The global structure of the sequence planner is illustrated in fig. 5.1. As described in chapter 2, the sequence planner works in two phases, the first product oriented, the second system oriented. The process in the left half of the figure (*Make_Clusters*) only depends on the structure of the product to be assembled, the processes in the right half (*Check_Stable* and *Expand_and_Verify*) also need information about the system that will perform the assembly task.

The first step in sequence plan generation (*Make_Clusters*) is to group parts that can be assembled as one group. These groups, or *clusters* as they are called, introduce the possibility to represent assembly plans more abstractly. More important, clusters make it possible to reduce the number of evaluations to be made in the search for the optimal sequence. Cluster generation is described in more detail in section 5.2.

In the second step (*Check_Stable*), stability heuristics are used to further reduce the number of sequences under consideration (see section 5.3). Potentially unstable part configurations can be made stable by using devices like clamps or special grippers.

In the last step of the sequence planning process (*Expand_and_Verify*) the still implicitly encoded sequences remaining after the last heuristic elimination step are expanded into a network structure that explicitly lists all sequencing alternatives for every stage of the assembly process (see section 5.4). To verify that each step is executable, the assembly of each part is expanded by lower level planners. The result of an expanded assembly step is a detailed plan for assembling the part (a part level plan), which includes a cost factor that combines estimates for the assembly cycle time and the chance of successful completion (see chapter 6).

5.1.3 The Input

As indicated in fig. 5.1 the input to the sequence planner contains three parts: the product model, a process model and a system model.

Figure 5.1: Global structure of the Sequence Planner.

The Product Model

The product model consists of three parts: the first part is the high level description of the product composition in the part list. An important difference with the Bill of Material (BOM) found in the lower right corner of a usual design drawing, is that for every part in the product, there is a unique identification.

This identification is needed in a second structure, the Part Position Network (PPN). The position of each part in space is described. No distinction is made between equivalent parts as was explained in section 4.2. Instead, the part list encodes unique combinations of part types and product positions.

The third part of the Product Model is the Part Relation Network (PRN). This network describes the global structure of the product: Parts are assumed to be rigid bodies, whose position and orientation can be completely described with six degrees of freedom. The composition of the product is modelled with a network of binary relations. A distinction is made between contact and non-contact relations. In the current prototype implementation three contact type relations Plane-Plane contact (PP), Cylindrical Fit (CF) and Screw Fit (SF) are predefined. These relations are described below. New relations types can be added easily.

Relations are defined in a goal oriented way, specifying the contact situation between the parts together with their relative degrees of freedom after assembly, plus the relative motion that is needed to connect both parts. In most cases a relation definition contains only two elements: An assembly motion matrix and a reference frame, describing both the contact situation and the motion. In some cases however, the assembly process is not reversible, e.g. when it is based on a plastic or elastic deformation (snap fit, bend). Here two separate sets of matrices and reference frames can be used.

The elements of motion matrix describe the amount of freedom in each direction of the orthogonal coordinate system defined by the motion reference frame (r_x denotes rotation around x):

$$\begin{pmatrix} x & -x & r_x & -r_x \\ y & -y & r_y & -r_y \\ z & -z & r_z & -r_z \end{pmatrix}$$

Predefined motion definitions:

f = Degree of freedom
 g = Guarded motion
 s = Screw motion
 0 = No motion allowed

Predefined relations:

1. The Plane Plane Contact

$$A \text{ PP } B : \begin{pmatrix} f & f & 0 & 0 \\ f & f & 0 & 0 \\ 0 & g & f & f \end{pmatrix}$$

Motion Reference Frame: Anywhere on the plane of contact, z -axis aligned with the normal vector of the contact surface of B, pointing towards A.

2. The Cylindrical Fit

$$A \text{ CF } B : \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ f & f & f & f \end{pmatrix}$$

Motion Reference Frame: Anywhere on the centerline of the peg, z -axis aligned with the centerline.

3. The Screw Fit

$$A \text{ SF } B : \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ s & s & s & s \end{pmatrix}$$

Motion Reference Frame: Anywhere on peg centerline, $-z$ -axis aligned with centerline, pointing in screw direction.

An example of a part relation network will now be given. Fig. 5.2 depicts the example product, the Cranfield Benchmark, an artificial product especially developed for testing robot programming systems by Cranfield Institute of Technology [RC84]. A detailed description of the product is provided in

Part no.	Code	Name	Unique
1	SP_1	Side Plate	No
2	SP_2	Side Plate	No
3	Peg_1	Spacer Peg	No
4	Peg_2	Spacer Peg	No
5	Peg_3	Spacer Peg	No
6	Peg_4	Spacer Peg	No
7	B_l	Spacer Block	Yes
8	S_h	Shaft	Yes
9	H	Handle	Yes

Table 5.1: Part list for the Cranfield Benchmark

chapter 8. For clarity, a simplified version of the Cranfield Benchmark is used here, consisting of nine parts (CB9).

Figure 5.2: The Cranfield Benchmark product.

The parts are listed in table 5.1. Fig.5.3 shows the corresponding relation network

Process Model

The one step in the assembly process that is the most critical for assembly sequence planning is the final insertion of parts into a semi-completed product. Therefore a Process Model of this step is included in the sequence planner while other detailed information processing like global motion planning and grip planning is delegated to lower level planners.

Figure 5.3: Relation network for the simplified Cranfield Benchmark (CB9).

In the sequence planner, the assembly process is modelled as a sequence of states, each state describing the status of all parts in the product. Each state indicates all relations that have been established with each part. In the sequence planner prototype only two final insertions (assembly motions) have been predefined: linear motion and spiral (screw) motion. This set can be easily extended. As an example the definition of the cylindrical Peg-Hole connection is presented, a combination of Cylindrical Fit and Plane Plane contact:

$$A \text{ PH } B : \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & g & f & f \end{pmatrix}$$

Motion Reference Frame: On the centerline of the peg, at the point where the "head" of the peg touches the surface. The z-axis is aligned with the peg centerline.

The Peg-Hole motion matrix can be computed from the CF and PP motion matrices by applying the following combination rules per motion direction:

$$0 + 0 = 0 \quad (5.1)$$

$$0 + f = 0 \quad (5.2)$$

$$0 + g = 0 \quad (5.3)$$

$$g + g = g \quad (5.4)$$

$$f + g = g \quad (5.5)$$

$$f + f = f \quad (5.6)$$

Similar rules have been defined for combining screw relations with other relations and for combining motions in different coordinate systems.

Partial assembly is not allowed. A relation between two parts is either fully established or not. Once a relation is created, it will not be broken until the product is complete. The planner does not require all relations of one part to be established at the same time. The process model does support subassemblies.

System Model

The sequence planner does not need a very detailed model of the system. Most details are left to lower level planners. One important exception is the concept of stability. A group of parts that is not stable by itself may be stabilized by a special multi-part gripper. An unstable element might be held in place temporarily by some clamping device. The system model assumes that stabilizing devices are designed with specific parts, part types, or situations in mind, and that potential stabilizing actions can be listed explicitly when appropriate.

5.1.4 The Output: The LASTD

To represent the output of the Sequence Planner, the Layered Assembly State Transition Diagram (LASTD) is used. As described in the previous chapter, the *nodes* in this representation represent the static states of the parts in the assembly. The *arcs* represent the assembly processes to get from one state into the next.

As described in the previous chapter, the LASTD representation was designed to be an interface between planning and scheduling, but is also used

during the planning process.

5.2 Clusters

In chapter 4 it was shown that a detailed examination of all assembly sequences can not be done with a trial and error method. For any reasonably sized product, there will just be too many combinations to be explored. The first step to deal with this complexity was to separate product and system oriented phases in process planning. However, both phases are still very complex. As a second step it is proposed here to exploit the mechanical structures in products, by introducing abstraction levels inside the product model. This new concept, called clustering, makes it possible to deal with sequences per group of parts which is much more efficient than dealing with single parts separately.

5.2.1 Definitions

A cluster is a group of parts that belong together logically because they have some assembly features in common, because of which they normally will be assembled together. The features in common relate to mechanical structures in the product. The combined parts do not necessarily form a stable subassembly. A key assumption is that once the assembly of a group is started, the sequence is not interrupted by parts from outside that group. Depending on the type of cluster, the sequence of assembly of the parts within the group may be fixed, or can be determined at a later stage.

In general, a cluster is a group consisting of two or more elements. Each element refers to a single part or to a child cluster (making the definition of a cluster recursive).

Child clusters may overlap, or in other words, there may be several ways to build the same product. Child clusters that do not overlap are grouped into a cluster tree under one parent, child clusters that have partial overlap are described as alternative reductions of that parent cluster.

There exist two basic types of clusters: stacks and layers. Stack Clusters group parts with fixed precedence relations between them. In general, there is only a limited number of sequences that guarantees successful assembly of the stack. Fig. 5.4 presents an example: Part *C* is *stacked* on top of part

B, thus *B* has to be assembled first.

a) Product

b) Relation Network

Figure 5.4: Example of a product with a stack cluster. *CF* = Cylindrical Fit, *PP* = Plane Plane contact, *SF* = Screw Fit. Parts *B* and *C* form a stack cluster

Characteristics of a Stack Cluster are:

- Each element has at least one contact relation (is in contact) with at least one other element in the group.
- All elements have a common assembly direction
- The group as a whole has only one (net) external contact relation, such that the group as a whole can be assembled without collisions. Elements in the group do not have contact relations to any elements outside.

Layer Clusters group elements with the same assembly direction but without fixed precedence relations. In general, many sequence alternatives are possible. Fig. 5.5 presents an example: Between parts *B* and *C* there is no sequence constraint.

Characteristics of a Layer Cluster are:

- All elements in the cluster have similar relations (a common assembly direction and similar relative positions) with element outside the group.
- Each element in the cluster has at least one contact relation with an outside element. Often the elements will have two "outside" relations: the layer elements are sandwiched in between.

Figure 5.5: *Example of a product with a layer cluster. CF = Cylindrical Fit, PP = Plane Plane contact. Parts B and C form the layer cluster*

Together with a disassembly approach to sequence planning (as described in the next section), stacks and layers are the basis for exploiting the physical structure of the product to reduce the combinatorial complexity of sequence planning. For increased performance (further reduction of complexity) new cluster definitions can be easily added. In the prototype sequence planner, two specialized cluster types were added: the Sort Cluster and the Combi Cluster.

The Sort Cluster is a special form of Layer Cluster: Each element is a single part, and all parts are of the same type. All parts in a sort cluster have similar relations (a common assembly direction and similar relative positions) with one or more parts outside the group. Fig. 5.6 presents an example: A lid (*B*) is fastened on a box (*A*), by means of two bolts (*C* and *D*).

The Combi Cluster combines characteristics of stack and sort clusters. It consists of several stack clusters that contain identical parts with similar (internal and external) relations. An example is shown in fig. 5.7: a set of bolt and ring stacks that attaches a cover.

To increase the flexibility of the prototype planner, and to be able to focus the attention of the sequence planner on a certain part of the assembly sequence, a possibility to predefine clusters (by hand) was added. During sequence planning the group of parts making up the user defined cluster is treated as a single part.

Figure 5.6: Example of a product with a sort cluster. *CF* = Cylindrical Fit, *PP* = Plane Plane contact, *SF* = Screw Fit. Parts *C* and *D* form a sort cluster

Figure 5.7: Example of a product with a combi cluster. *CF* = Cylindrical Fit, *PP* = Plane Plane contact, *SF* = Screw Fit. Parts *C*, *D*, *E* and *F* form the combi cluster

5.2.2 Cluster Recognition

Cluster recognition is an iterative process. After a quick check for sort clusters (done only once per product), the basic procedure is as follows: First the product model is checked for part groups that meet a cluster specification. Then the part relation network is reduced (clusters that were found are "eliminated") and the procedure is repeated until the solution becomes trivial.

The dataflow diagram in fig. 5.8 gives an impression of the procedure as it was implemented in the final prototype. This procedure has two predecessors. The first algorithm that was implemented used a "connective-element" approach, growing sequence groups from connective parts by successively adding and verifying related parts [Bon88], [BH89]. After one cycle of recognition is complete, the clusters found so far were deleted from the product model and the process was repeated. The method relied heavily on the definition of a connective element (e.g. bolt, shaft etc.).

The second algorithm [Hee89] used a pure disassembly approach, growing clusters from "removable" parts in the product model. Again, clusters found were deleted from the product model and the process repeated until the solution became trivial.

Comparison of both approaches [Guu89] learned that the disassembly approach produced somewhat better results with respect to the number of correctly found clusters. By working "bottom-up", the connective element approach had to examine many instable combinations (subassemblies) of parts, while the disassembly approach automatically ignored these combinations. The final prototype implementation is based on the disassembly method, but contains some improvements that were derived from the "connective element" approach.

One of the improvements is that the detection of sort clusters works directly from the part list. The part list is simply checked for multiple occurrences of the same part type. Each of these groups is then checked for corresponding assembly directions and corresponding relations with others.

Theoretically, combi clusters can be detected in two ways: as sort clusters that are stacked on top of each other, and as stack clusters that contain similar parts and have similar relations. The connective element approach used detection via sort clusters. A slightly different definition of clusters made this relatively "cheap" method (because sort clusters can be detected di-

Figure 5.8: Cluster recognition cycle

rectly from the BOM) useless in combination with the disassembly approach. Instead, combi cluster recognition was based on related stacks detection.

5.2.3 Theoretical implications of clustering

The clustering process can be regarded as a first step in the search for the shortest path (the cheapest sequence) in an enormous one level ASTD, representing all possible assembly sequences. The number of nodes in an ASTD is typically much smaller than the number of transitions. Deletion of one node will usually render several transitions useless, several transitions have to be eliminated before one node loses its meaning.

The effect of clustering can be measured in terms of the number of nodes in the LASTD and the corresponding number of sequences. Clusters combine nodes into sequence groups, that can be examined independently. This leads to a large reduction in the number of sequences to be expanded. Furthermore, the remaining sequences are much shorter, because they either describe the process at a higher abstraction level (in terms of clusters instead of single parts) or only for a limited number of parts (inside the cluster). In summary, the effect of clustering is based on the following principles:

1. The assumption that assembly of a cluster is non-interruptable eliminates many irrelevant sequences. This assumption is based on the definition of a cluster, which corresponds to physical structures in the product.
2. The introduction of abstraction levels makes that sequence constraints translate to more powerful decisions (elimination of a single high level transition can eliminate many sequences, see section 4.4).
3. The strong sequence constraints inside a stack cluster, leave open only a few alternative sequences. Again, the effect is based on the cluster definition, corresponding to physical structures in the product.
4. Within a layer cluster there are no fixed sequence constraints. Special heuristics can further reduce the number of sequence alternatives. The freedom of sequence is further exploited in a later phase of sequence planning (during expansion), to minimize the number of tool changes.
5. New cluster definitions can introduce other sequencing rules within clusters

<i>n</i>	<i>c</i>	<i>p</i>	<i>Seq</i>	<i>Nodes</i>
24	0	6	$6.2 \cdot 10^{24}$	$1.7 \cdot 10^7$
	1	6	$1.2 \cdot 10^{17}$	$5.2 \cdot 10^5$
	2	6	$8.7 \cdot 10^{10}$	$1.6 \cdot 10^4$
	3	6	$3.6 \cdot 10^5$	704
	4	6	2904	272

<i>n</i>	Number of parts in the product
<i>c</i>	Number of clusters in the product
<i>p</i>	Number of parts in a cluster
<i>Seq</i>	Number of assembly sequences
<i>Nodes</i>	Number of nodes in the LASTD

Table 5.2: *The effect of the number of clusters on the complexity of sequence planning*

Cluster size and number

It is difficult to provide accurate general estimates of the amount of problem reduction provided by clustering, because the effect depends on many factors: Cluster size, number of clusters, number of levels, overlap etc. In this section, two (theoretical) examples will provide some preliminary insight, a practical example (based on the Cranfield Benchmark) will follow in section 5.5.

A first, intuitive interpretation of the clustering concept will yield the notion that the effect depends on the number of clusters involved. This is indeed true, as is illustrated in table 5.2.

For a single level of c non-overlapping clusters, a total number of parts n , and a fixed cluster size (p) the number of (partial) sequences encoded (*Seq*) and the number of nodes (*Nodes*) are as follows:

$$Seq = (n - c \cdot p + c)! + c \cdot p! \quad (5.7)$$

$$Nodes = 2^{(n-c \cdot p+c)} + c \cdot 2^p \quad (5.8)$$

Equation 5.7 expresses that the total number of sequences equals the number

n	c	p	Seq	$Nodes$
24	2	12	$9.6 \cdot 10^6$	8196
	3	8	$1.2 \cdot 10^5$	776
	4	6	2904	272
	6	4	864	160
	8	3	$4.0 \cdot 10^4$	320
	12	2	$4.8 \cdot 10^8$	4144

n	Number of parts in the product
c	Number of clusters in the product
p	Number of parts in a cluster
Seq	Number of assembly sequences
$Nodes$	Number of nodes in the LASTD

Table 5.3: *The effect of cluster size on the complexity of sequence planning*

of sequences in the top level (the factorial of the number of elements) plus all sequences in the bottom level clusters. Equation 5.8 expresses the same for the number of nodes (remember that a full ASTD for a product with x parts contains 2^n nodes).

Table 5.2 presents the results for a reasonably complex product (total number of parts $n = 24$) and a cluster size of $p = 6$. The number of clusters (c) was varied from 0 (no clusters) to the maximum of 4 (all parts member of a cluster). The results clearly indicate that an increasing number of clusters results in a dramatic reduction of LASTD complexity: The number of sequences to be considered e.g. reduces from $6.2 \cdot 10^{24}$ to a mere 2904, without even applying any heuristics to reduce the number of sequences inside the clusters.

One should not conclude from this example that more clusters automatically implies a better result. The effect also depends on the size of the clusters involved, as is illustrated in table 5.3.

The table presents the results for the same size of product ($n = 24$) and a varying cluster size. The product is assumed to be fully clustered into a single level of non-overlapping clusters ($c = n/p$). The equations for the number of sequences and the number of nodes presented above still apply.

The results indicate that there is there is an optimum complexity (a low

<i>n</i>	<i>c</i>	<i>p</i>	<i>Seq</i>	<i>Nodes</i>
24	2	12	6	52
	3	8	12	56
	4	6	32	64
	6	4	732	112
	8	3	$4.0 \cdot 10^4$	304
	12	2	$4.8 \cdot 10^8$	4144

<i>n</i>	Number of parts in the product
<i>c</i>	Number of clusters in the product
<i>p</i>	Number of parts in a cluster
<i>Seq</i>	Number of assembly sequences
<i>Nodes</i>	Number of nodes in the LASTD

Table 5.4: *The influence of stack cluster size on the complexity of the LASTD*

number of nodes and a low number of sequences) when cluster size and cluster number are average. When there are only a few clusters, the internal complexity of these clusters dominates (the effect of a large *p*). When there are many small clusters, the complexity of the top level diagram of the LASTD dominates.

The sequence inside a stack cluster

The clustering approach is valid as long as the parts inside the cluster do have a tight relation (in a physical sense). The constraints on stack clusters (same assembly direction *plus* fixed sequence relation) are much stronger than the constraints on layer clusters (same assembly direction). This is reflected in the number of sequence alternatives that are available inside the cluster.

The definition of the stack cluster implies that there are only two sequences inside. The definition of the layer cluster implies no extra sequence constraints. The effect of the stack definition for a varying cluster size is illustrated in table 5.4.

The table presents the results for the same size of product as was used earlier ($n = 24$). The product is again assumed to be fully clustered into a single

level of non-overlapping clusters. Because there are only two sequences inside a stack cluster, the equations for the number of sequences and the number of nodes have to be modified:

$$Seq = (n - c \cdot p + c)! + c \cdot 2 \quad (5.9)$$

$$Nodes = 2^{(n - c \cdot p + c)} + c \cdot 2p \quad (5.10)$$

The results indicate that now the optimum complexity (the lowest numbers of nodes and sequences) occurs when cluster size is at maximum. When there are only a few large clusters, the internal complexity (in this case simplicity) of the stack clusters dominates. When there are many small clusters, the complexity of the top level diagram of the LASTD dominates the node and sequence count.

The sequence inside a layer cluster

Inside a standard layer cluster there are no physical sequence constraints. In some special cases heuristics may be applicable: E.g. when attaching a wheel to a car, the nuts have to be tightened in a special sequence. Together the nuts form a sort cluster, a special form of layer cluster. Also, in a later stage of sequence planning, the freedom of sequence inside a layer cluster can be exploited to minimize the number of tool changes (see sections 5.4 and 5.5).

5.3 Stability heuristics

Because of their definition, clusters encode for partial sequences. This alone leads to a significant reduction of the complexity of the sequence planning problem. However, there is a second advantage: with relatively simple heuristics the number of sequencing alternatives can be further reduced by eliminating unstable states involving clusters.

One example is, that at any time, all parts should be in contact with each other or with some stabilizing device. The result will be the elimination of one or several states from the LASTD. The effect of an elimination corresponds to the complexity of the situation. The elimination of the a certain

state can make several transitions meaningless. In other words, the deletions will propagate through the state transition network. Further examples of stability rules are presented in section 5.5.

5.4 Expansion of sequences

The last step in sequence planning is to expand the hierarchy of assembly sequences in the LASTD into a complete tree of single transition steps, to call lower level planners to compute the assembly cost for each part insertion and, finally, to search the optimal path through the tree. Now there are two problems: For each call, the lower level planners need information about preceding states, which depend on the assembly sequence chosen. This implies a combinatorial explosion in the number of branches in the search tree. The second problem is that the computing cost of generating one lower level plan can be quite high.

The expansion process tries to cope with these problems by using several techniques (see fig. 5.9). At the heart of this process there is a mechanism that selects the next transition in the ASTD to be expanded. The selection mechanism is based on the A* algorithm [BF81, Vol. 2, ch. C3] a heuristic search algorithm that, under appropriate circumstances, is guaranteed to find the optimal solution. One condition is, that in every situation considered, it must be possible to generate an optimistic estimate of the total cost (in this case: assembly time) to complete the whole product starting from that situation.

The quality of the cost estimates influences the quality of reasoning, i.e. when the estimates differ too much from the actual value, many alternatives have to be explored. In the current design, the cost estimates were based on Boothroyd's assembly time analysis [BPM82].

A second mechanism tries to look ahead one step to see whether a tool change will be required. The cost of a tool change is usually of the same order of magnitude as the complete insertion of one part. When possible, a tool change is avoided. Thus the number of steps that have to be expanded — corresponding to the number of lower level plans that have to be generated — is reduced.

Figure 5.9 depicts the two mechanisms as two loops: the loop on the right side selects and lets the part level planner expand transitions. The left

Figure 5.9: Sequence expansion

loop tries to look ahead one step and precomputes the cost estimates. Both loops may operate in parallel, transition by transition. When backtracking is applied, only the right loop is active.

5.5 Example: Cranfield Benchmark

This section gives an example of cluster recognition and the application of stability rules to create a sequence plan for the Cranfield Benchmark. To keep the example simple, a reduced version of the Cranfield Benchmark with 9 parts (CB9) is used. Table 5.1 contains the part list, fig. 8.1 shows the parts on the base plate, fig. 8.2 shows the complete assembly. Figure 5.10 shows the relation network. The theoretical number of sequences is $9! = 362880$.

Figure 5.10: Relation network for the 9 part version of the Cranfield Benchmark (CB9)

The first step is the detection of sort clusters. There are several candidates, as can be seen from the part list (table 5.1): The two side plates and the four spacer pegs are not unique. No sort clusters are detected though, because none of these parts can be removed as a group.

The second step is the recognition of layer clusters. Because all parts have the same assembly direction, the disassembly approach detects the whole product as a layer. This is not the result we hoped for, because the definition of the layer cluster leaves open every possible sequence.

With no useful clusters found in the first round, the sequence planning procedure has to fall back on the basic rule of disassembly: the relation

network is reduced with one "removable" element, and the cluster detection process is restarted. There are two candidates for reduction: Side Plate 1 and Side Plate 2. Figure 5.11 shows the relation network after reduction of SP_1 . A similar network exists with SP_2 reduced. The number of sequences under consideration is reduced to $2 * 8! = 80640$.

Figure 5.11: Relation network for CB9 after reduction of SP_1

The following cluster recognition cycle starts again with the recognition of sort clusters. This time, two clusters are detected, S_1 containing Peg_1 and Peg_2 , and S_2 containing Peg_3 and Peg_4 . The four pegs can not be grouped into a single four part cluster, because Peg_1 and Peg_2 have an additional relation with the spacer block B_1 . The combined relation of two pegs in each sort cluster with the side plate loses the rotational symmetry. In figure 5.12, showing the relation network after the reduction of SP_1 and two sort clusters, this is indicated with a new relation type: the SP-contact (square peg contact). Again, a similar relation network exists with SP_2 and the sort clusters reduced. Both relation networks contain 6 elements; both sort clusters contain 2 elements. Thus, the number (partial) sequences still under consideration is reduced to $2 * 6! + 2 * 2 = 1444$ partial sequences.

In the following step, reasoning from the remaining side plate, all parts are again recognized as one big layer cluster. Reasoning from the opposite side, one finds a layer consisting of all parts *except* the side plate. This last

Figure 5.12: Relation network for CB9 after recognition of two sort clusters

cluster receives priority, because it encodes new information. The result is that the LASTD obtains an extra level. The top level, shown in figure 5.13, describes all possible sequences for the two side plates and the layer parts in. The second level describes all (partial) sequences for the five constituting elements of this layer cluster (LA_1): the two sort clusters, the shaft, the handle and the spacer block. At the lowest level, the two sort clusters are further expanded.

The layer recognition step concludes cluster recognition. Inside the layer cluster there are no more relations between the parts that lead to new accessibility constraints. The top level of the LASTD encodes only four sequences because either SP_1 or SP_2 has to be assembled last (see figure 5.13). Inside the layer cluster there are (for now) no sequence limitations. The remaining LASTD thus encodes $4 + 5! + 2 * 2! = 128$ (partial) sequences.

Further reduction can be obtained with stability rules, e.g.:

1. A layer cluster may not be assembled as the first step in an assembly sequence.
2. If there is only one part that has a relation with a fixture, this part is assembled first.
3. A peg-hole relation proceeds plane-plane relation. Planar contact by itself is not stable. Gravity combined with friction give only a small

Figure 5.13: Top level LASTD for CB9 found with cluster recognition and the disassembly approach

resistance against disturbing forces (e.g. vibrations from the pallet transport system).

The first rule eliminates the middle intermediate state from the top level of the LASTD (see fig. 5.14), reducing the number of sequences at this level with 2.

Figure 5.14: Top level LASTD for CB9 after first stability rule (left), and after the second rule (right)

The second rule implies that part position SP_1 (the bottom part in the fixture) will be filled first. Note: Here we see an implication of using part positions instead of explicit references to parts: Assembling part position SP_1 first still leaves the choice for either one of the two side plates in the kit open! The result for the LASTD is that only one sequence remains available at the top level: One *has* to start with the bottom side plate (SP_1), followed by all the parts in (layer cluster LA_1), and concluding with the side plate on top (SP_2 , see fig. 5.14).

The third rule has some consequences inside the layer cluster at the second level (LA_1): All sequences that imply the sub-sequences Block (B_l) *before* sort cluster S_1 (containing the stabilizing Peg_1 and Peg_2) and Handle (H) *before* Shaft (S_h) are eliminated.

Figure 5.15 presents the final result: A three level LASTD. The total number of (partial) sequences that is encoded is reduced from 362880 to a mere 37. Further reduction is possible, e.g. based on the number of tool changes. For this last optimization a detailed knowledge of the available tools is necessary.

Figure 5.15: Total LASTD for CB9 after three stability rules. Further optimization is possible, e.g. based on the number of tool changes

Chapter 6

Part Level

The part level is the third level of the Assembly Planning Reference Model presented in chapter 2 (see fig. 2.1). The part level planner deals with one part at the time. For each part it selects an adequate set of resources and a set of feasible action sequences.

The part level acts as an interface between the abstract state changes that represent assembly actions at the product level, and the detailed real-world actions of the primitive level, that are tightly coupled to real actuators and sensors working on real objects.

To make optimization possible, the part level planner must keep track of the assembly cycle time. It must select the fastest sequence of actions, but it must also guarantee that the each step of that plan will work. A central element in this task is to keep track of all the uncertainties that influence the geometric relations between the parts that make up a product.

The first section of this chapter describes the global structure of the part level planner. The second section describes how geometric relations in the assembly environment influence the assembly process. The third section presents a model for describing geometric relations and associated uncertainties. The fourth section goes deeper into the computation of critical relations, which are playing a key role in the decision making of the part level.

6.1 Part Level Planning

6.1.1 A script based approach

The part level planner has to translate abstract "assemble part" commands, generated by the product level, into device specific primitive level actions. As there are many types of devices that can be integrated in the assembly environment, the best basis for the part level planner is a generalized strategy that can be easily extended with new primitives. Here, a script-based approach is proposed.

The basis of the approach is the Part-script. The Part-script consists of a series of primitive actions. Each primitive action is linked to a fixed (set of) resource(s), e.g. a tool, a robot or combination of these. Primitive actions can be combined into clusters (macro-primitives) that improve script readability and ease of use.

Primitives come in two types: planning and action primitives. A planning primitive (e.g. *Select_robot*) selects the most promising resource from a set of competing resources. If this resource later turns out not to be applicable (because the associated uncertainties are too high), the planner will come back to this step and select an alternative resource.

There are three types of action primitives:

estimate - *< primitive >* This procedure provides a "quick and optimistic" assembly cost estimate for the associated primitive action. Depending on the desired amount of detail, values can be assigned to macro-primitives only, or to more detailed primitives actions. The whole set of estimation procedures is run only once for each part. The value it generates is based on global part characteristics, and is used to guide the global direction of search in the Part-script.

simulate - *< primitive >* During planning this procedure simulates the execution of the associated primitive action, using all information gathered using previous planning steps. The result is a more accurate (and usually somewhat less optimistic) assembly cost estimate, that further guides the search. The procedure also updates the internal model of relations and uncertainties, to represent the new (expected) state of the assembly environment.

execute_ < primitive > During program execution this procedure makes the appropriate calls to the associated device.

The ordering of primitive actions is explicitly described, eg. *a ; b* implies that action *a* precedes action *b*. Two primitives can be joined by an OR link, indicating that both primitives are more or less equivalent. The planner will select the cheapest alternative, but can backtrack to another selection when the chosen one turns out not to work.

Figure 6.1: Graphical representations of the top-level Part-script (a), and part level plan generation (b).

Figure 6.1-a gives a graphical impression of the top-level part-script. Because of the modular structure, the scripts can be easily re-arranged or extended to include new primitives for other devices.

6.1.2 From script to plan

The part level planner searches the Part-script for the optimal part plan (with a short cycle time and a reliable result), using an algorithm similar to the ASTD expansion algorithm used at the product level. The global structure of the part level planner is depicted in fig. 6.2.

The central element in this flow diagram is the Part State Transition Diagram (PSTD), a network-like data that describes all primitives that have been or can still be examined. The planner iteratively updates the PSTD, identifies the best (macro-)primitive to expand, expands the Selection-primitives (selects a resource), checks the accumulating uncertainties and computes new estimates.

The cost factor (C) that is used to guide the search is based on two factors: The time it will take to execute the action (T) and the probability (P) that the action will be successful:

$$C = T/P \quad (6.1)$$

The search process expands the network like structure of the Part-script into a search tree. The result of a single pass is a linear plan. Multiple passes can extend it to a part-plan with alternatives. An example of the creation of a linear part-plan is depicted in fig. 6.1-b.

The plan shows that the first attempt to grasp the part did not work out. The planner uses backtracking to go back to the point where an alternative action is available. In this case, including a sensor improves the knowledge of the parts position enough to make the succeeding grasp and mount operations successful.

The rest of this chapter describes the concepts for representing geometric relations and dealing with accumulating uncertainties in the assembly environment.

6.2 Geometric relations in an assembly environment

To assemble a part correctly, several geometric relations between the part and the product have to be established, e.g. the centroid of a peg has to

Figure 6.2: Global structure of the Part Level Planner (ProMOD dataflow diagram)

be aligned with the centroid of a hole and the bottom of the peg must be aligned with the bottom of the hole. It is the task of the assembly system to create these part-product relations. Spatial uncertainties in the system will influence its capability of creating accurate relations. For automatic action planning, both product and environment must be modeled.

A convenient concept for describing relations between objects in space is the frame concept. Each object is assigned a reference frame. Relative positions of objects are expressed as relations between these reference frames. Objects can have associated features, that are linked to the local reference frames. Features make it possible to use high level abstractions for describing the relations between the objects.

Figure 6.3 gives an impression of a frame based description of the assembly environment. In the figure, only a small number of objects was assigned a reference frame. The model that is actually used in the part level planner contains much more detail. An important assumption for the frame concept to be valid, is that all objects are rigid bodies.

6.2.1 Existing approaches

In literature, several approaches can be found that attempt to describe assembly environment at a similar abstraction level.

AUTOPASS was one of the earliest attempts to build a more intelligent assembly system. It was developed by Lieberman and Wesley at IBM Yorktown Heights Laboratory and was called a "task level" programming language [LW77]. The AUTOPASS world model could describe several types of relations between parts and tools: Attachments, motion constraints, and features (with part-of relations). AUTOPASS could not describe uncertainties. The sequencing of actions was done by a user who defined assembly actions explicitly in terms of GRASP and MOUNT commands (defined as primitive level actions here). AUTOPASS was never fully implemented.

Another, related approach is the "object level" language RAPT, developed at the University of Edinburgh [PAB78], [PAK82]. RAPT is an abbreviation for Robot APT. Like APT (a language used in parts manufacturing), RAPT concentrates on formalized specification of the spatial relations between parts. Because these relations can be established by primitive level actions (robot moves) the link to robot programming is obvious.

a) Assembly environment

b) Corresponding network of frames

Figure 6.3: *Modelling the assembly environment with a network of frames*

A typical RAPT statement is e.g.:

```
move/plate, perpto, side of plate;  
    against/side of plate, locsurf of block;  
    aligned/phole, bhole;
```

Figure 6.4: A typical action that can be specified in RAPT [PAK82]

In the RAPT formalism this means about the same as: *Move the plate in the direction perpendicular to the side of the plate, such that the side of the plate is against the locating surface of the block, and the hole in the plate is aligned with the hole in the block*, describing a sliding motion of the plate over the block. A preceding guarded motion has ensured contact between plate and block (see fig. 6.4).

RAPT models objects with the APT formalism, i.e. with points, lines and faces. A user supplied, explicit description of the desired spatial relations between objects is used to infer robot motions. In recent versions of RAPT, the mechanism that reasons about spatial relations is powerful enough to deal with data from several types of geometry sensors like range finders. The specification of spatial relations between geometric features of parts can be very useful for planning grips on parts or mounting strategies.

For part level planning, a description of spatial uncertainties is thought to be essential. The RAPT system is still under development, but according to available information, no description of spatial uncertainties is included. This specific area is further explored in the next sections.

6.2.2 The nominal position of an object

In general, a rigid body in space has six Degrees Of Freedom (DOF), and its exact position and orientation can be described with six independent parameters.

The goal of the assembly process at the part level is to create a relation between the part and the product by establishing the correct values for all position parameters (the term position parameter is assumed to include orientations). The way such a value can be established depends on the characteristics of the position parameter. First consider the nominal value of a single position parameter (corresponding to a single degree of freedom):

Irrelevant The value of the parameter is irrelevant. Example: The orientation of a cylindrical peg around its axis of symmetry.

Constant The parameter has a constant value that can be extracted or computed from information available during part level planning. Example: The height of a table.

Variable The parameter has a value that is variable in a general sense, but constant for each specific instance of an object. The value is not known exactly during part level planning, but can be computed from information coming available during execution of the assembly program. Example: In the assembly cell of fig. 6.3, part A is brought into the working area of the robot on a belt feeder. After the belt has stopped, the position of the part remains constant, but its value cannot be predicted exactly.

Dynamic A parameter is dynamic when its value changes continuously while other (related) actions take place. Example: In the assembly cell of fig. 6.3, part A is brought into the working area of the robot on a belt feeder. Now, the belt keeps on moving during part pick-up and the robot has to adjust its motion dynamically to the motion of the belt.

Combining these four parameter types over six DOF creates many types of position relations. They can be grouped into two broad categories:

1. Position relations containing only parameters whose value can be completely determined during planning. Except for an occasional irrele-

vant orientation of a symmetrical part, all parameters are of the constant type.

2. Position relations containing one or more parameters that are variable or dynamic. The complete relation can not be computed before actual execution.

The first category covers the relations describing the position of objects that are fixed in all DOF by mechanical means. Many relations describing the system set-up fall in this category. An example is the position of a presentation point of a dedicated parts feeder (e.g. a vibratory bowl feeder), with respect to the table it is mounted on. Provided that all relations are known with sufficient accuracy, the planner can exploit this information to base its plan actions on dead-reckoning.

The second category covers relations that describe more dynamic processes. The frequency of change depends on the type of action involved. Some relations remain constant over a longer period of time (e.g. the position of a tool with respect to the gripper changer), other relations are subject to frequent changes (e.g. the position of the robot TCP).

Relations of the second category are often needed to describe the position of a part when it is introduced into the assembly environment, e.g. when it is fed in a bin or on a transport belt. The position of the bin or the belt is usually known, but other position parameters have to be measured. Knowledge about stable orientations of parts and knowledge about overlapping conditions might help in recognizing a part in a certain situation.

To deal with relations of the second category, the planner assumes that predefined strategies are available that can compute the correct parameter values provided the rough position of the object is within certain bounds. Through propagation of known (or simulated) spatial uncertainties, the planner checks the preconditions.

6.2.3 The Propagation of Spatial Uncertainties

The assembly environment is not perfect. Due to manufacturing tolerances, tool wear, finite stiffness, calibration errors etc, there will always be some uncertainty about the exact position of objects.

In the assembly environment there are many sources of uncertainty:

The product The dimensions of parts are subject to manufacturing tolerances.

The system System components are also subject to manufacturing tolerances. In many cases several components are stacked on top of each other (e.g. a gripper on a robot, the robot on the floor), which can amplify small inaccuracies into large position errors.

The assembly process Many actions essential to the assembly process directly or indirectly influence the quality of the relation that can be established between part and product. The introduction of a new part into the assembly system is a major source of uncertainty. Some feeding approaches have an inherently large uncertainty (e.g. bin picking). The assembly process produces also some side-effects that feed back into the accuracy of future actions, e.g. wear can decrease a tools grasping accuracy.

External disturbances Some sources of uncertainty lay outside the scope of the assembly system. Dimensions of parts and system components will vary according to environmental temperature and humidity. Vibrations can introduce dynamic errors in the relative positions of system components.

A major problem in creating an accurate description of part level actions, is that the success of one action depends on the success of the previous action. Assembly actions have different capabilities in terms of positioning accuracy or spatial information gathering. In other words: assembly actions can accumulate or reduce uncertainties and propagate them to there successors.

Figure 6.5 presents an example of accumulating uncertainties. When two objects A and B with uncertain dimensions are stacked on top of each other, the height of the resulting assembly AB has a larger uncertainty. The figure illustrates a worst case approximation, other models for accumulating uncertainties are presented in the following sections. Objects A and B could be parts, but they might also represent a feeder on a table etc.

From a part's point of view, there are three major steps leading to the goal (being assembled): FEED, GRASP and MOUNT. Upon introduction of a part in the assembly environment (FEED), mechanical tolerances of the feeder system are combined with uncertain part dimensions into an uncertain feed position. Inaccuracies or uncertainties that occur in any DOF after

Figure 6.5: An example of accumulating uncertainty in the assembly environment. Here a deterministic (worst case) approximation is illustrated: The uncertainty in the size of objects A and B combines into a larger uncertainty for the size of assembly AB.

completion of the feeding step must be reduced to acceptable values before grasping, unless the parameter value is irrelevant.

Figure 6.6: A sensor can reduce uncertainty in the assembly environment. As long as the object is within the range of the sensor, its position can be measured. The accuracy that can be achieved depends mainly on the characteristics of the sensor.

Figure 6.6 presents a simple example of active uncertainty reduction: A sensor measures an object's position. As long as the object is within the range of the sensor, its position can be measured with an accuracy that mainly depends on the characteristics of the sensor. Reduction of uncertainty can also be achieved by mechanical means: e.g. by pushing a part in an accurately defined corner or using a self aligning gripper. Both types of uncertainty reduction are supported in the prototype planner.

Goal of the GRASP primitive is to acquire a stable grasp on the part. Important factors are part geometry, gripper geometry, the way the part is

fed (limiting the accessible sides) and the way it should be assembled (again limiting the accessible sides of the part). Selecting a grip and a gripper, generating an approach trajectory to the pick-up point and synchronization of the grasp action with robot motion and the part feeder are all tasks that are dealt with in detail at the primitive level. Task of the part level planner is to keep track of the accumulating uncertainties and to link in a sensor operation or ask the lower level planners to select a different gripper when the (simulated) result of the grasp operation indicates failure.

The gripper that is used to grasp the part is mounted on a tool changer, which in turn is mounted on the robot arm. Small deviations in gripper alignment or robot mounting can be amplified into a large position error by moving the part over a large distance. The geometric model in the planner incorporates this accumulation effect.

To MOUNT the part, it is transferred along an optimal path from a point near the position where the part was grasped, to a position that is a good starting point for the final insertion. The insertion itself is primarily a relative positioning problem, comparable to part grasping. The position of the part in the gripper can now assumed to be known, as should the position of the product in which we are inserting our part be. An unknown product position is treated in a way similar to finding the position of a part before grasping.

Grasp and mount actions can reduce the uncertainty about a parts position, through passive compliance in gripper or robot. Also stabilizing fixtures can be introduced that actively or passively (through gravity) improve the knowledge of the parts position. The effects of mechanical compliance and alignment are included in the geometric model used in the part level planner through explicit specification of pre- and post conditions.

6.2.4 Sensor interaction

In an assembly environment, many types of sensors can be used. Most relevant to the part level are position sensors. Depending on range, accuracy and the number of dimensions in which one wants to know an object's position, the choice ranges from simple binary mechanical switches and inductive, capacitive, or optical presence detectors, via force-torque sensors based on strain gages to sophisticated vision systems.

Another factor that complicates the use of (position) sensors in an assembly

environment, is that usually the sensing actions have to be synchronized with other actions. Position sensor - actuator interactions can be classified into three groups:

1. The value is measured with some sensor as a sequential step in the program. Example: A part is brought into the assembly environment on a belt feeder. The belt stops for part pick-up. The position of the part now is constant but will have an arbitrary value within a certain range. As a first step one can ask a vision system for the part coordinates. The second step is to compute the desired gripper position from the vision information, and as a third step the manipulator moves the gripper to this position (Look-And-Move).
2. The parameter is measured with some sensor while the action is performed. The output of the sensor is checked continuously (by a parallel process) against preset boundaries (Guarded Action). If a boundary value is reached, the execution of the main process is interrupted. Example: An instrumented gripper with independent fingers can adjust itself to the exact position of a part by stopping the closing motion of a finger at the moment it touches.
3. The parameter is measured with some sensor while the action is performed. The (continuous) output of the sensor is used as an input for some control loop (Adaptive Control). Example: Tracking a part on a conveyor belt.

Table 6.1 presents some examples for each sensor information processing strategy, applied to sensors of different complexity.

The geometric model in the part level planner allows sensor measurements to be incorporated into the relation network as newly created relations between a sensor frame and the measured part. In the planner prototype, sensor interaction primitives are structured like other "low-level" primitives: device specific procedures written in the C-language (see chapter 7).

Signal complexity	Processing strategy		
	Sequential Action	Guarded Action	Adaptive Control
Binary e.g. - micro switch - photo cell	Presence check Simple synchroni- zation	Presence guard	Simple level control
Parametric e.g. - strain gage - Ultrasonic transducer	Look and Move in 1 DOF Level measurement	Search in 1 DOF Insertion force monitoring	Path tracking in 1 DOF
Complex e.g. - tactile array - vision system - torque/force sensor	Look and Move in multiple DOF Part identification	Search in multiple DOF Part slippage monitoring	Path tracking in multiple DOF Active compliance

Table 6.1: *Examples of sensor information processing strategies for various types of sensors*

6.3 Modelling the Assembly Environment

6.3.1 The nominal position

Assuming all objects (parts, tools, etc.) are rigid, the relative positions of objects in the assembly environment is represented as a network of spatial relations between features of the objects. Features are convenient points of reference, assigned to each object. One object can have multiple features (e.g. a block can have two holes). Each feature is represented by a coordinate frame. The relative position of two features on one object and the relative position of two objects in space are described by the same type of relations (e.g. the relative position of two holes in a single block is described in the same way as the relative position of a peg in a hole).

A relation is defined as the transformation from one frame (the reference frame) to another (the goal frame, see fig. 6.7). In this thesis, right handed orthonormal coordinate frames are used.

The transformation between reference frame and goal frame has two components: A translation vector describing the position offset and a description of the orientation difference between the two frames. Usually one of the

Figure 6.7: Definition of a relation

following three methods is used to describe orientation differences:

- Algebraic parameters
- Trigonometric parameters
- Direction cosines

The methods are compared in [Ron88]. The three methods differ in compactness, the time needed to compute transformations from one frame to another, and use of memory.

For user convenience, the part-level planner accepts a trigonometric representation as input for describing orientations. There is no problem with singularities, because only stable orientations are described. To propagate uncertainties, the part planner does have to compute inverse transforms, and singularities can pose a problem. Therefore, inside the planner, the orientation angles from the input are converted into a direction cosine representation (a 3×3 matrix). This is not the most compact form, but its concept is easy to understand and it can be elegantly combined with the translation vector into a single, easy to handle 4×4 matrix known as the homogeneous transform [Sny85].

6.3.2 Uncertainty and Approximate Transforms

In the assembly environment there is usually some uncertainty about the exact values of position parameters. Many of these parameters can not be accurately described with a nominal value alone. However, often it is possible to describe the probability that a specific value will occur with a distribution function.

When the number of position parameters is small, joint distribution functions can accurately represent all information on resulting errors. In the assembly environment however, there are many interdependent relations, and some approximation must be used. Some researchers model the entire environment with normal (Gaussian) distribution functions [SC86]. This model is not always accurate (it is sometimes too optimistic), a.o. because part dimensions and feed positions tend to contain systematic errors, which cannot be described with normal distributions alone. Others take a worst case approach and describe all uncertainties with fixed errors bounds [Bro82], [TR88]. The worst case approach often leads to rather conservative results [Dri88], [Baa89].

Careful analysis shows that the choice between a conservative worst case and an optimistic stochastic approach is not the only issue to be addressed in modelling the assembly environment. The following (interdependent) modelling problems have to be dealt with:

- How to represent the uncertainty distribution for a single parameter?
- How to represent the combination of position parameters in a single relation?
- How can relations be combined?

Uncertainty distribution of a single parameter

To be useful for planning, the model should provide a fairly accurate model of the assembly environment. To keep the model simple, we would like to represent all relations with one standard distribution function, that could cover all relations.

The stochastic approach models all uncertainties with normal (Gaussian) distribution functions. Systematic errors are assumed not to be present.

The worst case approach assumes no knowledge of the distributions involved. The uncertainty is represented with a lower and upper bound only. The real value of the parameter can be anywhere between both bounds. Multiple instances of the same parameter may show a systematic error. As systematic errors occur quite often in the assembly environment, the worst case description seems more useful.

Combining position parameters in a single relation

A single relation between two parts has six interdependent components. For decision making, the part level planner must combine these components and compare the combined distribution with some norm values. To keep the representation simple, we would like to express the chance that the combined distribution of n position parameters is be equal to or less than a specific value in a simple way.

Figure 6.8: *Conservative approximation of dependent position parameters in 2 DOF. The "real" uncertainty (solid box) is a function of both x and y . Modelling both parameters to be independent makes it necessary to approximate the uncertainty area (dashed box).*

One possible assumption is that all parameters in one relation are independent. Combining relations becomes a simple operation, as is shown in the next section. A disadvantage of the independency assumption is that rather conservative approximations are needed when the parameters are not independent. An example is shown in fig. 6.8.

Combining relations

To describe spatial uncertainties the standard frame concept is extended to a network of *Approximate Transformations (ATs)*. An AT consists of an *Estimated Mean (EM)*, a homogeneous transformation describing the nominal relation and an *uncertainty descriptor (U)*, in the form of a vector or matrix. The assembly environment is modeled as a network of AT's (see figure 6.9). Every AT describes nominal relations and uncertainties in six degrees of freedom. The uncertainties are symbolized by rectangles in the figure.

= Critical path

□ Critical relation (to be determined)

Figure 6.9: The assembly environment modeled as a network of AT's

The success of a part level assembly action depends on the accuracy of the relations that it establishes. To check the quality of a plan, one has to reason about the relations that represent the key to success, e.g. the relation between the Gripper and Part A in figure 6.9. Such a key relation is called a critical relation.

Figure 6.10: Comparing the uncertainty of a critical relation with a norm. In this case the relation between gripper and part is compared with the opening range of the gripper. The relative position of the part is not known accurately enough and the grasp operation is bound to fail.

The value of the critical relation is computed by propagating known spatial uncertainties through the *AT*-network. The net value is then compared with a predefined norm value, which depends on the local geometry, and the abilities of the tools involved. Figure 6.10 gives an illustration: The relative position uncertainty of the part (critical relation) is larger than the opening range of the gripper (norm). The planner concludes that the plan needs improvement.

The nature of the comparison depends on the type of distribution involved. Figure 6.10 illustrates a conservative (worst case) approximation, where the comparison of uncertainties leads to a black-or-white result: The uncertainty of the critical relation either does or does not match the norm. Uncertainties with a Gaussian distribution will never completely match a norm. Instead, they result in a chance that a match will occur.

6.3.3 Example

Before going deeper into the problem of computing the critical relation, the propagation of uncertainties through the *AT*-network is illustrated with an

example. The top of figure 6.11 shows the initial network, representing an assembly cell with a robot, a tool rack containing a camera and a gripper, and a belt feeder. Now, a part is fed on the belt. In the internal representation, a new relation is created between part and belt. The relation has a large uncertainty. Following the Part-script, the planner selects the robot and the gripper (not a very difficult choice in this simple example).

Action primitives simulate the robot taking a gripper from the tool rack (top) and attempting to grasp the part on the belt. Unfortunately, the uncertainty in the position of the part is too large (bottom of the figure). The planner uses backtracking and goes back in the part script. A sensor (the camera) is selected. Action primitives now simulate the robot taking a camera from the tool rack and measures the position of the part (top of figure 6.12). The measurement results in a better known position of the part.

Following action primitives simulate the camera being restored in the tool rack and the gripper being taken from the tool rack. Using the measured position, the grasp operation now succeeds (bottom of figure 6.12). The corresponding steps in plan generation are depicted in 6.1.

Figure 6.11: *Propagation of position uncertainties 1: The robot takes a gripper from the tool rack (top) and attempts to grasp the part on the belt. Unfortunately, the uncertainty in the position of the part is too large (bottom).*

Figure 6.12: *Propagation of position uncertainties 2: The robot takes a camera from the tool rack and measures the position of the part (top). The camera is restored in the rack and the gripper is taken up from the tool rack again. Using the measured position, the grasp operation succeeds (bottom).*

6.4 Computing a critical relation

Two ways for computing the critical relation have been explored:

1. The critical path method
2. The network reduction approach

The critical path between two frames in a network of AT's is the path that has the lowest net uncertainty. The accumulated uncertainty of this path corresponds to the uncertainty of the critical relation. In figure 6.10 there is only a single path from the gripper to the part. In a practical environment there may be multiple paths (e.g. multiple sensor readings. Then, the critical path could be found by a graph search algorithm.

Knowledge of the critical path could simplify decision making when the net uncertainty is too large: Sensor measurements could e.g. be included to improve the worst relations on the path. However, the critical path approach has a problem: Only in a one-dimensional world (as the one illustrated in fig. 6.10), there is a single critical path for every two frames. In the real world however, there are six position parameters to be dealt with. Interdependency between positions and rotations makes critical path computation very difficult or very conservative. Therefore, part level planner development was focussed on the network reduction approach.

6.4.1 The basics of network reduction

The network reduction algorithm repeatedly combines two AT's into one until the entire network is reduced to single AT (which is the critical relation). The algorithm propagates uncertainties in all six DOF concurrently. The algorithm uses five basic steps [vdDH89]:

1. Compound
2. Merge
3. Eliminate a Dead-end
4. Reduce a Loop
5. Invert a relation

Compound and merge are the actual network reduction steps, the other three solve situations that obstruct further reduction.

Compound

Compound combines two adjacent ("serial") relations into a single relation. Figure 6.13 gives an example. Relations AT_1 and AT_2 are combined into a new relation AT_3 . The compound action can be executed when an intermediate frame (a frame that is not the start or finish frame) takes part in exactly two relations. The intermediate frame (frame B) is eliminated.

Figure 6.13: *Compound of two relations*

Merge

A merge action combines two parallel AT 's. An illustration is presented in figure 6.14. A merge action can be executed when there are two parallel transformations between two frames. When there are two relations in anti-parallel, one of the two must be inverted first. In contrast to a compound step, no frame is eliminated here.

Figure 6.14: *Merge of two relations*

Deadend elimination

A dead-end in a network of ATs is a frame that is not the start or finish frame, and that takes part in only one relation. An example of a network with a dead-end is depicted in figure 6.15. The net relation between the start frame and the finish frame is not influenced by a dead-end. Therefore, a dead-end can be eliminated without any computations.

Figure 6.15: *Eliminating a dead-end*

Reduction of loops

Most networks can be reduced with the three basic operations described in the previous paragraphs. Sometimes however, a situation as depicted in figure 6.16-a occurs. No compounding or merging can be done, and no dead-ends can be eliminated. This situation is called a *loop*.

Using the worst case approach, the loop problem can be dealt with easily. The first step is to translate the irreducible situation into an equivalent, but reducible situation, as depicted in figure 6.16-b. This partial network can now be reduced with merging and compounding.

Inversion

A problem that will occur frequently is that transformations do not point into the right direction. This can be solved by inverting the AT, as depicted in figure 6.17. No reduction of the network is accomplished, but the invert operation is required to continue the reduction.

Figure 6.16: Reduction of a loop structure with the worst case approach. The network with the irreducible loop (a) is transformed into an equivalent, reducible network (b).

Figure 6.17: Inverting a relation

6.4.2 Implementation

Although the general concept of compounding and merging is fairly straightforward, their implementation can be complex, depending on the type of distribution function involved. First the deterministic (worst case) compound and merge of independent parameters will be described.

Deterministic Compound

A deterministic (worst case) compound action consists of two steps, i.e. computing the net nominal transformation EM and computing the net uncertainty U . The estimated mean is represented in a 4×4 homogeneous matrix, the uncertainty vector is represented as a vector, describing (symmetrical) uncertainties in the six main directions (x, y, z, r_x, r_y, r_z).

Let

$$AT_i = (EM_i, U_i) \quad (6.2)$$

where

$$AT_1, AT_2 = \text{two adjacent } AT's \quad (6.3)$$

$$AT_3 = \text{the net } AT \quad (6.4)$$

$$U_{ij} = \text{the } j^{\text{th}} \text{ element of } U_i \quad (6.5)$$

Then

$$EM_3 = EM_1 \cdot EM_2 \quad (6.6)$$

$$U_{3j} = U_{1j} + U_{2j} \quad (6.7)$$

An example in two dimensions is depicted in fig. 6.18. In this two dimensional case, the homogeneous matrix describing the nominal values (EM) is a 3×3 matrix. In this example, there are no rotations. The start situation:

$$EM_1 = \begin{pmatrix} 1 & 0 & 20 \\ 0 & 1 & 30 \\ 0 & 0 & 1 \end{pmatrix} \quad (6.8)$$

$$U_1 = (d_x, d_y, d_{r_z}) = (5, 5, 0) \quad (6.9)$$

Figure 6.18: Two dimensional example of a deterministic compound

$$EM_2 = \begin{pmatrix} 1 & 0 & 30 \\ 0 & 1 & 20 \\ 0 & 0 & 1 \end{pmatrix} \quad (6.10)$$

$$U_2 = (10, 5, 0) \quad (6.11)$$

Using the compound rules given above, the result is:

$$EM_3 = \begin{pmatrix} 1 & 0 & 50 \\ 0 & 1 & 50 \\ 0 & 0 & 1 \end{pmatrix} \quad (6.12)$$

$$U_3 = (15, 10, 0) \quad (6.13)$$

Deterministic Merging

Like compounding, merging consists of two steps: The net EM transformation and the net uncertainty vector have to be computed. Because of the worst case approach, deterministic merging is simple mathematically. Let

$$AT_i = (EM_i, U_i), \quad (6.14)$$

where

$$AT_1, AT_2 = \text{Two parallel } AT's \quad (6.15)$$

Figure 6.19: Example of a deterministic merge in two dimensions

$$AT_3 = \text{The net AT} \quad (6.16)$$

$$EM_{ijk} = \text{The } k^{\text{th}} \text{ element of the } j^{\text{th}} \text{ column of } EM_i \quad (6.17)$$

$$U_{ik} = \text{The } k^{\text{th}} \text{ element of } U_i \quad (6.18)$$

Also, let

$$UPPER_{3k} = \text{MIN}(EM_{13k} + U_{1k}, EM_{23k} + U_{2k}) \quad (6.19)$$

$$LOWER_{3k} = \text{MAX}(EM_{13k} - U_{1k}, EM_{23k} - U_{2k}) \quad (6.20)$$

Then:

$$EM_{3jk} = (EM_{1jk} + EM_{2jk})/2 \quad (6.21)$$

$$U_{3k} = (UPPER_{3k} - LOWER_{3k})/2 \quad (6.22)$$

An example in two dimensions is depicted in fig. 6.19. Again, there are no rotations. The start situation:

$$EM_1 = \begin{pmatrix} 1 & 0 & 30 \\ 0 & 1 & 45 \\ 0 & 0 & 1 \end{pmatrix} \quad (6.23)$$

$$U_1 = (d_x, d_y, d_{r_z}) = (10, 10, 0) \quad (6.24)$$

$$EM_2 = \begin{pmatrix} 1 & 0 & 45 \\ 0 & 1 & 30 \\ 0 & 0 & 1 \end{pmatrix} \quad (6.25)$$

$$U_2 = (10, 10, 0) \quad (6.26)$$

Using the merge rules given above, the result is:

$$EM_3 = \begin{pmatrix} 1 & 0 & 37.5 \\ 0 & 1 & 37.5 \\ 0 & 0 & 1 \end{pmatrix} \quad (6.27)$$

$$U_3 = (2.5, 2.5, 0) \quad (6.28)$$

Worst case merging is only valid if there is an overlap between the two areas ($UPPER_{3j} > LOWER_{3j}$ for all j). If the boxes describing the uncertainties of AT_1 and AT_2 do not overlap, there is no solution.

Stochastic approach

The stochastic approach proposed by Smith *et al* [SC86] models all uncertainties by normal distributions. The dependencies between the 6 position parameters of each relation the assembly space are recorded in a 6×6 matrix of variances and co-variances. Compound and merge operation become complicated matrix operations [Baa89]. To simplify computation, the trigonometric elements describing the dependencies between orientation and position in the transformation matrices are approximated by the first order Taylor series expansion. Therefore, only small angular uncertainties can be described.

In a later work Smith, Self and Cheeseman describe a global relations model, where dependencies between relations are also available [SSC88]. This method leads to less conservative estimates for situations where relations are interdependent, but the computational complexity is large when many relations are involved.

Combined Approach

Comparing deterministic and stochastic approaches as described in literature, shows they both have advantages and disadvantages. The main ad-

vantage of the deterministic approach is its simplicity. A problem is that the position parameters in the assembly environment are not independent. An approximation is necessary. On the other hand, the stochastic approach tends to be too optimistic, because not all relations can be accurately described with normal distributions. The stochastic approach however, provides a better model for repeated measurements and repeated moves, etc.

As part of the work described in this thesis a new approach was evaluated that combines elements of the deterministic and stochastic approaches. Several types of uncertainty distribution functions are supported, so one can choose the distribution function that best matches reality. All six position parameters of a single relation are assumed to have the same type of distribution.

Three distribution functions were implemented in the prototype planner: Normal distribution (e.g. suited for sensor measurements), uniform distribution (e.g. to approximate tolerances on part dimensions) and worst case (for unpredictable situations that may contain a large systematic error, like the feed position of a part on a tray). All three distribution functions are translated into an internal representation of uncertainties in a variance / covariance matrix.

A limited dependency between the position parameters within one relation is assumed. The multidimensional area that describes the joint distribution of the six position parameters within a single relation is approximated with an (six dimensional) ellipsoid. This choice seems to match well with natural phenomena (the ellipsoid represents the area of constant chance for a jointly normal distribution).

The mathematics of compounding and merging of a variance / co-variance representation are described in [Baa89]. Both deterministic and stochastic methods are used, depending on the type of relations involved. Only when both input relations have a normal distribution, the stochastic approach is used. In all other cases the deterministic approach is used.

When the number of relations increases, the worst case approximation of the deterministic approach gives more and more conservative results. According to the Central Limit Theory, the combination of independent relations of arbitrary distribution (containing no systematic errors) will always approximate a normal distribution. Assuming that the relations in the assembly environment are more or less independent, that relevant uncertainties are of the same order of magnitude and that systematic errors are unrelated, the

worst case approximation can be relaxed. In the prototype implementation, the combination of more than five relations is assumed to result in a normal distribution, implying that the less conservative stochastic approach can be used.

Chapter 7

Primitive Level

The primitive level is the fourth and lowest level of the Assembly Planning Reference Model presented in chapter 2 (see fig. 2.1). Primitive level planning deals with one primitive at the time. The technical applicability of available resources and strategies for using those resources are evaluated.

The actions at the primitive level are tightly coupled to real actuators and sensors working on real objects in the assembly environment. The primitive level planner optimizes resource selection by establishing assembly cost estimates for every action, and offering alternative primitives at a lowest cost (fastest device) first basis.

Because primitive actions are device specific and there are many types of devices that could be integrated in an assembly environment, the planner must adopt a modular approach, in which the primitives are treated as independent units. In the design described here, precedence relations between primitives are explicitly encoded in the part level Part-script.

As stated in chapter 6, a distinction is made between two types of primitives: planning and action (physical) primitives. Each planning (or selection) primitive is a kind of mini-expert system that selects the most promising resource from a set of competing resources. If the device later (in the simulated process) turns out not to be successful, the part-level planner can backtrack to the appropriate selection step and request an alternative resource. To speed up the planning process, a planning primitive may pre-compile a list of resources in the order of decreasing expected performance.

Because an introduction of primitives and their cohesion in part level scripts

was already presented in chapter 6, this chapter has a library-like structure. This structure corresponds well to the device specific, but modular and extendible nature of primitives.

Section 7.1 presents the selection primitives: `Select_feeder`, `Select_robot`, `Select_grip` and `Select_sensor`. The second and third sections describe the action primitives. For clarity and ease of use, several often re-occurring groups of low-level primitives are grouped in so-called macro-primitives. Section 7.2 describes the three macro-primitives of the basic pick-and-place cycle (as seen from a part's point of view): `Part_feed`, `Part_grasp` and `Part_mount`, and two examples of macro-primitives that combine low-level primitives in guarded actions (`Extended_line_scan` and `Vision_three_point_measure`). Section 7.3 describes the operation of some of the lowest-level, device bound primitives.

7.1 Selection primitives

This section describes the selection primitives: `Select_feeder`, `Select_robot`, `Select_grip` and `Select_sensor`. In the prototype implementation and the experiments, most emphasis has been put on grip selection. This primitive is therefore described in more detail.

7.1.1 `Select_feeder`

Supported feeding strategies are: Kit feeding, pallet feeding, belt feeding. Feeder selection not really poses a process planning problem. Experience shows that there is a clear ranking of these three strategies, independent of the part type: Kit feeding is faster than pallet feeding, which in turn is faster than belt feeding (ranking confirmed by experiments described in chapter 8). Selection of the best feeder can therefore be done by working down the list. Note that this selection is based on cycle time estimates inside the workstation only. Here, kit feeding is so fast because the distance between feeding and mounting positions is so small. Overhead outside the workstation is not included (e.g. the time needed to fill a kit). A fair comparison of the "real" cost can in this case only be done at a higher level (not considered in this thesis).

7.1.2 Select_robot

A robot is selected on basis of its speed, working area (are both start and goal location within reach?) and the number of degrees of freedom that is required. Each single part is only handled by one robot.

7.1.3 Select_Grip

Goal of the Select_Grip primitive is to find the right gripper and gripping strategy that will get a part under control. Important factors are part geometry, gripper geometry and the way the part is fed (limiting the accessible sides).

Whether a certain type of gripper is suitable to grasp a certain part largely depends on specific part features: usually parts are grasped on parallel sides or on cylindrical surfaces. Some grippers can grasp a part on a single surface (vacuum, magnetic grippers).

To be able to clarify the procedure of grip selection, some terms have to be defined:

Grasp A step in the assembly process

Grip A unique combination of a part feature and a related gripper posture

Griprimitive A configuration of part or gripper features that can be used to create a grip

The complexity of grip selection increases with the number of possible gripper/part combinations (grips). In current industrial systems there is only a limited set of grippers available. The geometry of the gripper fingers is usually especially designed for one part type (dedicated gripper), for a small set of part types (multi-feature gripper), or for a limited range of parts (e.g. with a rectangular or cylindrical cross-section). Also, most grippers have simple actuating and control mechanisms (two rotating or translating fingers activated by OPEN and CLOSE commands), that make them only suited to limited range of part sizes. Thus, the number of griprimitives that is supported by the gripper is restricted.

Several laboratories (Stanford/JPL, UTAH/MIT, Hitachi) are working on human like hands, with three or four fingers and many degrees of freedom.

Griprimitive	Nr. of fingers	Object surface
1.	Special purpose	One flat surface (vacuum, magnetic)
2.	Two finger	Two parallel flat surfaces
3.	Two finger	Cylinder vertical
4.	Two finger	Cylinder horizontal
5.	Three finger	Two parallel flat surfaces
6.	Three finger	Cylinder vertical

Table 7.1: Griprimitives supported in the planner prototype

One might expect that such a universal gripper would eliminate the grip selection problem. In fact, the reverse is true: The universal gripper does eliminate the problem of selecting the right gripper, because the same gripper can be used for all parts; On the other hand, the universal hand offers a large range of grip configurations, using two, three or four fingers in planar, tripod or other configurations. Instead of reducing the problem, the number of alternative griprimitives supported by the gripper tends to increase. Multi-DOF hands are not further considered.

The global structure of the grip planner is depicted in fig. 7.1. A central role is played by the griprimitives. Together they form a rough classification of elementary grip configurations and can be computed for parts and grippers separately. Table 7.1 gives an overview of the griprimitives supported in the planner prototype.

First all possible griprimitives for each part are determined. Then, these griprimitives are checked for collision with any feeder or product surfaces. The (relative) position of these collision prone areas is already known from previous planning steps: The relative mount position is computed at the product level, the contact surfaces between part and feeder are decided by the Select-feeder primitive.

The griprimitives that remain are called free-griprimitives. If no gripprimitive is found this is reported by the bad-part flag as no-griprimitive.

The next step is to choose a possible gripper from the set of available-grippers. Each combination of chosen gripper and gripprimitive forms a supported-grip. This selection is based on general criteria. If no gripper can match any of the given griprimitives the bad-part flag is set to no-gripper.

The last step is to examine the supported-grips and sort out the good grips.

Figure 7.1: Global structure of the grip planner

If no good grip can be found, the `bad_grip` flag is set to `no_grip`. First, the remaining grip configurations are evaluated by a set of evaluation rules. An example of such a rule is "Distance-to-COG", a procedure that computes the distance between the center of the grip to the Center Of Gravity of the part. In this case, the grip rating is inversely proportional to this distance.

For each rule, the ratings of the grips are compared with limitations set by the specific part or gripper, e.g. whether the eccentricity of the grip is not so large that the part will slip, rotating between the fingers. Finally, the ratings are weighted and the best of the remaining grips is selected.

Similar to the structure of the sequence planner at the product level, there is a first phase of grip selection that can be executed independent of the assembly system. All possible gripprimitives for each part can be determined right after a product is designed. In a similar way, all possible gripprimitives for a gripper can be determined right after gripper design, independent of the parts that are to be assembled.

7.1.4 Select_sensor

In practical assembly environments only a limited number of sensors will be available. The majority will be of a simple binary type. The application of more complex sensor types (vision, taction) is increasing. In the primitive level planner, strategies for sensor use are assumed to be incorporated into pre-programmed procedures. The main task of the `Select_sensor` primitive is to check the preconditions for sensing (right part, working area etc.)

7.2 Macro primitives

This section describes macro-primitives, fixed sequences of low-level primitives, that have been grouped for convenience and clarity. First, the three actions that constitute the basic pick-and-place cycle (as seen from a part's point of view) are described: `Part_feed`, `Part_grasp` and `Part_mount`, followed by two examples of macro-primitives that combine several low-level primitives in guarded actions (`Extended_line_scan` and `Vision_three_point_measure`).

7.2.1 Part_feed

When parts are fed in a kit, or on a pallet, they are in fixed positions. The Part_feed primitive computes the position of the next available part. This is done with the same environment model as is used in part level planning to (net) compute uncertainties. The only interesting information for the Part_feed primitive here is the nominal position of the part. When we imagine all parts in place on the feeding tray, we see a large tree of part frames all having the feeding tray as a reference. All that has to be done by Part_feed primitive is to activate one of these part frames.

7.2.2 Part_grasp

The Part_grasp macro primitive consists of many steps: In the first step it checks whether the robot contains the right gripper. If the robot is empty or holding a different gripper, the macro primitive Exchange_tool is called: In case of a tool exchange system (TES), Exchange_tool calls the macro primitive TES_grasp. If the robot is holding a different gripper, the macro primitive TES_release is called first to restore that gripper in the tool rack. In case of a multi-gripper Exchange_tool calls the macro primitive Index_multigripper. In the second step, Robot_move (7.3.1) is called to bring the part to the approach point. Then the gripper is opened with the primitive Gripper_release and the part is approached with a straight line motion (another call to Robot_move). Finally the gripper is closed with Gripper_grasp and the robot gently retracts with the part to the approach position (another call to Robot_move).

7.2.3 Part_mount

The Part_mount primitive consists of two steps: First Robot_move is called to bring the part to the approach point. Then the part is inserted with a straight line motion.

The final insertion of the part is primarily a relative positioning problem, comparable to part grasping. The position of the part in the gripper is assumed to be known. The product in which the part is inserted has six degrees of freedom. If the position of the product is not known accurately enough, the part level planner can backtrack to a product-sensing primitive before part_grasp. Two approach strategies are supported: Straight line

vertical approach and combined vertical and horizontal approach to insert parts horizontally.

7.2.4 Extended_line_scan

A first example of a macro-primitive that combines a sensor and actuator in a guarded move is the extended line scan. The line scan camera has a limited field of view. The robot is used to extend it with a guarded motion: The camera checks for edges while the robot is moving. The procedure involves three system elements (robot, camera and synchronizer — part of the workstation controller), and proceeds as follows:

1. The robot is instructed to move to the start position.
2. When the robot has arrived (the synchronizer is notified by the robot-generated "Robot-Moving" signal falling back to its default level), the camera is instructed to start looking and to generate a state output when an edge is detected.
3. The robot is instructed to start the scanning motion, and to stop on a "Robot-Stop" signal from the synchronizer. The synchronizer is instructed to pass a camera generated "Edge-Detected" signal on to the robot as "Robot-Stop" (extra work when this case is considered in isolation, but a contribution to an open structure in a larger context, see chapter 8).
4. After the robot has stopped, the actual robot position and the position of the edge in the view of the camera are read. Via a known pixel to millimeter conversion factor and the known offsets of the camera with respect to the robot (found by calibration), the position of the edge in robot coordinates is computed.

7.2.5 Three point measurement of part position

Another example of a more complex macro-primitive is the three point measurement of the position of a belt fed side plate (one of the Cranfield Benchmark parts, see chapter 8). The procedure consists of three extended line scans, as described in the previous section. First, the robot moves to the first start point (A_s) and starts measuring towards the first goal (A_g). The

Figure 7.2: Measurement of side plate position with three line scans

edge is detected in position A . At this point robot and camera positions are read. The robot moves to the second start point (B_s) and restarts the search etc. From the combined measurements of robot and camera edge positions in A , B and C , the position and orientation of O' can be computed.

7.3 Low-level, device oriented primitives

This section describes the operation of the smallest building blocks of the primitive level: the low level, device specific action primitives. Some of these primitives (e.g. to activate a tool) are very simple, merely generating a short string of ASCII characters. Other primitives (e.g. Robot_move) are more complicated, requiring a complete list of parameters.

In the two testbed sites (for a more detailed description see section 8.4), the low level primitives have been implemented in two different ways, corresponding to two different workstation control philosophies, and illustrating the advantage of the information hiding approach.

Details of signal levels, signal duration, communication channels and protocols are all hidden in library of "device driver" subroutines, written in a high level language (C). Because the implementation of these primitives is device specific, some of the details in the following paragraphs apply especially to the two testbed sites that have been developed for the experiments. It should be clear however, that the high level language basis of the primitives ensures that the implementation can be easily extended to other sites.

7.3.1 Robot_move

Goal of the Robot_move primitive is to bring a robot along an optimal path from one specified position to another, e.g. to transfer a part from the position near to where the part was grasped, to a position that is a good starting point for the final insertion. Important issues are: Generation of a collision free trajectory, the interface to a real robot, and synchronization of the robot motions with actions of other devices.

Trajectory generation

Many motion planning algorithms are described in literature [Ram86]. Important criteria to select the best algorithm to use in a certain situation are:

- Does the path have to be *the* shortest path or is a *a* short path sufficient? For flexible assembly the emphasis is on automatic generation of motion. When motion generation is executed off-line planning time is less important and a little more effort can be easily spent to generate somewhat shorter paths. For on-line motion generation, planning time has to be balanced against execution time. Finding a short path will usually suffice.
- How accurate does the path have to be? Off-line planning may spend lots of time on finding a fast but still safe path. If motion planning is done on-line, planning time has again to be balanced against the speed of execution (a low accuracy implies a large safety distance). In assembly there is a clear distinction between "fine" motions near or in contact with objects (e.g. approach, insertion) and "coarse" motions in a roughly free space (e.g. between the feeder and the product fixture).
- How many degrees of freedom are there in the problem? Many algorithms work only in low dimensional spaces. For assembly with a single robot the algorithm should be able to handle at least 4, but preferably 6 DOF.
- Is a guarantee needed that the path will work? Local path planning algorithms may be fast, but only global algorithms will not become trapped in a local optimum. In assembly we would like the part indeed to be assembled. Therefore, global algorithms are preferred.
- What resources are available to compute the trajectories? Some algorithms put high demands on computation time and memory use.

The problem of trajectory generation can be easily treated as a separately solvable one. Also considering the enormous amount of work that has already been done and is still being done by other research groups, this area was not given high priority. Instead, a simple, heuristic trajectory generation algorithm was developed. The modularity of the primitive level ensures that "better" trajectory generation algorithms are substituted easily.

The heuristic trajectory generation algorithm is a combination of two basic rules: Up-and-over and Via-point-to-area. The first rule exploits the fact that in many assembly systems there is free space everywhere above a certain height (especially in systems built around SCARA-type robots). The rule gives all trajectories the same basic form: Straight up from the start location onto the "free height", in a straight horizontal line to a point above the goal, and then straight down to the goal.

The second rule defines simply structured, pre-defined areas of interest in the horizontal plane, e.g. a box-like area around a tool rack. Each of these areas has an explicitly defined entry point. When moving from outside to inside such an area, the trajectory generator automatically inserts a via point (intermediate point) in the trajectory.

The interface to the robot

Many state of the art robots have advanced controllers with "high level" programming languages. Apart from the robot motions, they are often used to control the complete workstation controller as well.

On the other hand many robots lack possibilities in their basic motion repertoire: Guarded motions can only be implemented in a rudimentary form, compliant motion is often not supported at all. Most robots (even very simple ones) however, have some basic functions with which guarded and compliant motions can be emulated: Binary inputs, and an IF-THEN like construct in the programming language that allows conditional jumps on input changes. Part of the work therefore involved the development of a basic robot driver to emulate complex motions, via a standardized interface to higher levels of control. In the experiments, two approaches were evaluated:

For the Asea site a "traditional" approach was chosen, where the robot controller also acts as workstation controller. Activation of the grippers and the tool changer, and synchronization of robot actions with the outside world (e.g. the pallet transport system) are controlled from within a single, off-line generated robot program in a robot specific language (ARLA). Accordingly, in the planner, primitives like Robot_move or Gripper_grasp generate one or more lines of robot language source code in the off-line program.

In the ROVIS site a distributed control approach was chosen. The robot controller was degraded to a "motion interpreter", concentrating on robot moves. Devices like the transport belt and the locking pin rotation unit, but

also the tool changer on the robot and the tool mounted in the changer, are treated as independent, asynchronous devices, controlled via separate communication channels. Accordingly, in the planner, primitives like `Robot_move` or `Gripper_grasp` generate device-specific messages to be sent over device-specific communication channels.

Synchronization with other devices

For synchronization there is again a difference between the two control concepts. In the off-line approach (applied in the Asea site), most of the synchronization is explicit in the sequence of statements in the robot program. Only for synchronization with the transport system, binary signals are used.

In the on-line approach (implemented in the ROVIS site), synchronization of robot motion takes place via three binary signals: `Robot_start`, `Robot_moving` and `Robot_stop`. `Robot_start` is an input signal that tells the robot that it may start the next motion. It can e.g. be used to synchronize the robot with the gripper: To wait with moving until the gripper is fully closed. `Robot_moving` is an output signal that is set by the robot while it is moving. This signal can e.g. be used to synchronize the gripper with the robot: Wait with closing the gripper until the robot has come to a stand-still. `Robot_stop` is a signal used in guarded motions. When set, the robot should stop as quickly as possible.

7.3.2 Sensor actions

Many sensors in the assembly environment are of the simple binary type, delivering binary signals. Examples of such simple sensors are a light gate to detect part presence on a transport belt, and a gripper mounted part presence sensor (can also be implemented as a light gate). More complex sensors can deliver flows of information. An example of such a sensor is a line-scan camera, that can e.g. compute the position of an edge. Like the other low level primitives, details about signal duration and communication channels are hidden in "device driver" subroutines, written in C.

In the off-line approach, sensor signals are directly used in the robot / workstation controller program (e.g. "wait until input 1 = 1"). The on-line approach offers a more open and flexible environment because all signals between devices are routed via the workstation controller. Synchronization

conditions are (have to be) more explicit. It is easier to support the exchange of more complicated sensor signals.

7.3.3 Tools

Many devices in the assembly environment are controlled through simple binary signals. Like the other low level primitives, details about signal duration and communication channels are hidden in small "device driver" subroutines, written in C. Examples of such subroutines are `Gripper_grasp` (to close a gripper) and `Belt_start` (to start a transport belt).

Chapter 8

Experiments

Parallel to the development of the computer aided process planning concept, many hard- and software experiments have been carried out. During the first years the assembly program generation problem was approached from the assembly robot's perspective [Hee87b]. Correspondingly, experiments in the laboratory were focussed on robot programming.

As experimental results became available, the emphasis slowly shifted towards a product oriented approach. It became clear that the assembly process involves a lot more than just "going through the motions". As a result of the thought processes provoked by the experiments, the four level reference model was created. A first formalized description was presented at the first European Symposium on Assembly Automation in Veldhoven, 1987 [Hee87a].

To keep in touch with reality and to demonstrate the concept's feasibility, a prototype of the planning software has been built. Furthermore, because the planning software assumes the availability of a flexible assembly environment, two test sites were developed that could execute the newly created process plans.

The first section (8.1) describes the robot programming experiments carried out in 1986 and 1987. Section 8.2 introduces the experiments with the Planner prototype software and the testbeds. Section 8.3 describes the development of the planning software, followed by a description of testbed system selection and development in section 8.4. The main body of this chapter is formed by sections 8.5 to 8.8, successively describing the experi-

ments executed at each of the four levels of the planning concept and their results. The chapter concludes with an evaluation of the experimental results.

8.1 Experiments on robot programming

During the first years, experiments were focussed on program generation for assembly robots. In the first experiments it was examined whether it was possible to introduce the principles of structured programming in off-line robot program creation. A second series of experiments compared the "power" of several robot programming environments. A new benchmark product (DATE) was created to test specific assembly actions in more detail.

8.1.1 Structured programming

As test-product for the structured programming experiments, the Cranfield Benchmark was selected. This test-product was specially designed for benchmarking assembly programming systems by Cranfield Institute of Technology [RC84].

The Cranfield Benchmark product consists of 17 parts: Two side plates, four spacer pegs, a spacer block, a shaft (pivot), a handle and eight locking pins. Fig. 8.1 shows the parts as they are mounted on the benchmark base plate in the initial configuration. The task is to assemble all parts one by one into the assembly nest. A typical assembly sequence starts with mounting one of the side plates in the nest, and putting four spacer pegs in the appropriate holes. Then follow the shaft, the handle, the spacer block and the other side plate. Finally the securing pins are inserted. Figure 8.2 shows the expected result.

The structured programming experiments took place in the Robotlab and were performed on an ASEA IRB-1000 robot, programmed in the ARLA language [Mei87]. The tests revealed some limitations of the Benchmark design. Most were minor problems and improvements were easily made. One example is the over-constrained feed position of the side plates: The side plates were positioned by three round pegs, a statically over-constrained situation that sometimes caused the parts to jam during pickup. Results improved after one of the locating pins was removed.

Figure 8.1: Parts of the Cranfield Benchmark on the pallet. Top: Medusa model. Bottom: In the ROVIS site.

Figure 8.2: *The Cranfield Benchmark product in assembled form (Medusa model)*

One feature of the Benchmark turned out to be more important than expected: The variation of part dimensions. This necessitated the development of new grippers and illustrated the importance of stable finger design. On the other hand, the uncertainty in the feed position of the handle did *not* cause any problems, because of the self aligning capability of most grippers.

The structure of the product led to a program consisting of many repetitions of one basic pick-and-place operation, which was observed as a limitation of the benchmark design. Much time had to be spent teaching the correct positions, while little extra insight was gained. The use of sensors was not enforced by the product.

The experiments did lead to valuable insight into several aspects of the robot and the programming system, e.g. the complicated work envelope of the IRB-1000 robot, combined with the size and shape of the multigripper, led to reach problems.

A highly readable, structured program for the complete Benchmark product was created successfully. Compared to the effort for a "quick-and-dirty" version however, the effort needed to generate the structured program was great. The main reason is that the ARLA language is not very programmer-friendly. Subroutine calls take place by numbered labels, parameters can only be passed via registers, and there are no named variables except positions. The external axis was not supported off-line. In structure the language more resembles a macro-assembler than a high level language like Pascal.

Robot	Language	PGT	CTT	Stat	ACT
ASEA IRB-1000	ARLA	66	17	45	14
ASEA IRB-6/2	ARLA	54	14	69	18
IBM-TTT	AML	115	11	110	25
IBM-SCARA 7535	AML/E 3.1	42	14	58	20
IBM-SCARA 7545	AML/E 3.2	47	19	51	22
IBM-SCARA 7575	AML/2	60	8	69	15
PGT Program generation time (minutes)					
CTT Time for teaching coordinates (minutes)					
Stat Number of statements in program					
ACT Assembly cycle time (seconds)					

Table 8.1: Programming time for the four peg test

8.1.2 A comparison of robot languages

A second series of experiments was initiated to study the influence of robot programming environments on program generation time. The experiments were based on a small part of the Cranfield Benchmark: One side plate and four pegs. The task was to assemble the pegs into the side plate. The side plate was already in the nest, only the pegs had to be handled. Although limited in concept, this test did contain several basic assembly tasks: Pick-and-place actions, gripper control, linear interpolation, a repeat loop and subprogram calls. A fifth spacer peg feeding position was added to make it possible to test binary I/O and simple exception handling schemes.

The experiments covered a large number of programmers and six different sites, some in cooperation with robot vendors. The intention of the test was easily understood and the test could be performed rather quickly. The mechanical simplicity of the test provided little difficulty with finger design or robot reach. However, large differences in experience between programmers made programming time a bad measure for programming system performance. Table 8.1 shows the results after correction for programmer experience (i.e. by eliminating the results for inexperienced programmers).

The results indicate that the test was most easily programmed on a SCARA robot using the simple language AML/E [Ram87]. The language AML on the IBM-7565 gantry robot, generally observed as a very "powerful" lan-

guage, suffered from a slow editor and language complexity. A common element in all experiments was that a large portion of programming time was spent on teaching exact positions. The fastest way of teaching was shown on the IBM-7575 robot: The robot was put in a "limp mode" and manually guided to the appropriate locations (similar to playback programming).

8.1.3 DATE

The results made clear that the "four-peg-test" did not cover many aspects of assembly robot programming (e.g. there was only one direction of insertion, no rotation around the insertion axis was required). A new benchmark product, Delft Assembly TEst (DATE) was therefore developed (see fig. 8.3). DATE was designed to maintain the conceptual simplicity of the four-peg-test while covering a larger range of assembly oriented tasks, including some actions that required the use of advanced sensing strategies. To execute the test, a programmer had to adapt a basic pick-and-place task. Program *adaptation* time was taken as a measure for programming system performance [Lui88].

The benchmark product was designed to be solid, cheap and simple in construction. The main part is an aluminium block that is clamped onto a base plate. Small bars that are placed between the block and the base plate can be exchanged to provide small, well-defined position or orientation changes. The main assembly task consists of several pick-and-place operations. Three types of fit have to be made: A wide tolerance fit, a tight tolerance fit (0.06 mm); and a snap fit connection. Some holes were placed under an angle, to test robots with more than 4 DOF.

A first set of experiments was conducted by experienced programmers on three different sites. DATE appeared to meet most of the expectations [HL88b]. By that time the four level concept began to take form. It became clear that DATE, by concentrating on actions that are directly influenced by robot control, is best applied to test primitive level actions.

In the light of the four level concept, the small variations in the repetition of the basic pick-and-place cycle, identified but further ignored in the first experiments with the Cranfield Benchmark, were suddenly observed to correspond to significant actions at the higher abstraction levels: Small variations in mounting positions correspond to stacking parts on top of each other, increasing uncertainties and establishing relations between parts. The

Figure 8.3: *DATE: The Delft Assembly TEST product*

sequence of the basic actions is not fixed but neither is it arbitrary: Only a restricted number of variations is allowed, corresponding to the structure of the product. Thus, for testing higher level assembly actions the Cranfield Benchmark once again played an important role.

Apart from use in primitive level experiments, DATE has played another important role: It was used in the selection of assembly robots for the DIAC project. A first selection was based on the list of criteria that is part of DATE. Experimental assembly of the DATE product supported the final decision for one of the three remaining competitors [MH88], [Hee88a], [HM88].

For robot programming, a new approach was developed, in which the task of the robot controller was reduced to executing motions, and motion parameters are generated off-line (see section 8.4.3).

8.2 Experiments on process planning and plan quality

Experiments on process planning have been executed at all four abstraction levels. Some experiments concentrated on the operation of the planning algorithms, that were bundled in a prototype process planning package called the Delft Assembly Process Planner, or DAPPER. Other experiments concentrated on the quality of the (expected) result: assembly process plans. To study the applicability of the plans generated and to determine the influence of plans on the assembly cycle time, two test sites were developed, both built around industrial standard assembly robots.

The following characteristics of process planning and process plans were demonstrated and verified in the experiments:

A: Operation of the process planning algorithms at each of the four abstraction levels:

1. At the Batch level: The influence of strategical batch level decisions (strategies for parts feeding or tool exchange, the layout of the product transport system) on the assembly cycle time was examined. The Cranfield Benchmark product was assembled using different strategies.
2. At the Product level: The capability of automatically generating assembly sequence plans from a product description was studied. Correctness of the sequence plans and efficiency of plan generation were checked for the Cranfield Benchmark and for several other products.
3. At the Part level: The propagation of position uncertainties for real products in a real assembly environment and the sequencing of primitive actions were evaluated. As an example, one of the Cranfield Benchmark side plates was fed with high inaccuracy on a belt.
4. At the Primitive level: Correct operation of all primitives in general was demonstrated by the successful assembly of a complete benchmark product. To examine the operation of a selection primitive in more detail, grips were generated for many different parts. Detailed operation of several more complex action primitives (guarded motion, complicated measurement) was tested with one of the benchmark side plates. The ease with which new devices can be introduced (and the

Planner can be extended with new primitives) was examined by adding a turning station for locking pins.

B: Limitations of the four level planning concept

It was assumed that the system has been carefully engineered, resulting in a truly flexible assembly environment. The batch level experiments demonstrate that system layout decisions have important consequences on the assembly cycle time.

C: Flexibility of the planning approach

1. Flexibility of the approach with respect to systems is shown by assembling the same product (the Cranfield Benchmark) on two different sites (Asea and ROVIS site).
2. Flexibility with respect to products is demonstrated at the product level, by automatically generating assembly sequence plans for the Cranfield Benchmark and several other products. At the primitive level, flexibility is illustrated by assembling a part of DATE.

D: Optimization of assembly plans

The search strategies at the product and part levels plan guarantee generation of optimum assembly plans by a consistent application of assembly cycle time based evaluation functions in A* (A-star)-like search algorithms. Manual evaluation showed that indeed the best sequences are chosen. At the primitive level plan optimization implies offering candidate devices that can execute a requested primitive on a best (fastest) device first basis. Correct operation of the grip selection primitive was verified by manual evaluation of selected grips for many parts.

The following sections provide a more detailed description of the experiments and results. Table 8.2 provides a quick reference.

8.3 Software Development

8.3.1 Approach

Software development for the prototype Planner was focussed on those areas that so far had received little attention from the research community: As-

Section	Topic	Tested characteristic							
		A1	A2	A3	A4	B	C1	C2	D
8.3	Software Development								
8.4	Testbed Development					X			
8.5	Batch Level Exp.	X				X	X		
8.6	Product Level Exp.		X					X	X
8.7	Part Level Exp.			X			X		X
8.8	Primitive Level Exp.				X		X	X	X

Table 8.2: *Structure of this chapter*

sembly sequence planning (the product level) and the influence of uncertainties (the part level). The planning algorithms were bundled in a prototype process planning package that was named Delft Assembly Process Planner, or DAPPER.

In an early stage of the project it was decided that the prototype of the Planner software would be implemented on a MicroVAX 2 computer in the C language. Mainly for reasons of standardization, C was chosen as basis for the whole Planner. C provided a degree of flexibility that could not be offered by most existing robot programming languages. "Low level" software like robot-drivers and the link to the database were created with relative ease.

During the analysis and design phases, a structured approach was followed. Extensive use was made of a software package called PROMOD, which runs on an IBM-PC under MS-DOS and is based on the Yourdon methodology [HP87], [GEI87].

Using this structured approach had several positive effects:

- The documentation generated with the tool (especially the dataflow diagrams and the data tree) provided an effective means of communication between project members.
- The documentation made it possible to separate program design from code generation.
- Consistency of the data dictionary was automatically enforced.

The tool PROMOD has had its main value as an intelligent editor of dataflow diagrams. The PC version of PROMOD was slow with graphics, analysis, file I/O, and text editing functions were limited. This made using the tool very time consuming.

After tests were completed, the dataflow diagrams and supporting texts created during the design phase, were updated and adjusted. As expected, at the higher levels little adjustment was needed, at the lower levels sometimes an implementation with a different structure was chosen, requiring more than cosmetic changes in the design documentation. The end result is a set of well documented diagrams, making the software easy to transfer.

8.3.2 Database link

The first Planner prototype modules took their input from files. As a step to the multi-user, multi-tasking environment of the DIAC project, a link to a relational database was established. The Ingres Relational DBMS was selected as the DIAC standard.

Putting the data into a database did not lead to many new insights. Because a limited range of (simple) examples was used, the data was easily generated by hand and stored into a file.

The standard update facilities of INGRES (Query By Forms and Interactive SQL) provided some extra convenience for maintaining the data and some automatic verification of new input, at the cost of extra work on the interface and a reduced execution speed. The current effort should be seen as an investment that will pay-off when larger amounts of data make the management of file based data more difficult and error-prone.

8.3.3 CAD-link

Early experiments (with a prototype of the product level sequence Planner), showed that the creation of an interface to an industry-standard CAD system was not straightforward. An attempt was made to create a link to the Medusa package running on a Sun Workstation. 3D Models of all Cranfield Benchmark parts, the base plate and the completed product were created.

The only information that was obtained with relative ease was a geometric description of the parts, in the form of lists of points, edges and surfaces.

Even this information was not complete, e.g. cylindrical shapes were approximated with flat surfaces, and tolerance information was only available in text form. An accurate description of the relations between parts (a key element for product and part level planning) was very difficult to obtain.

An experiment with an experienced technician showed that a normal set of (part and composition) drawings does contain enough information to turn a stack of loose parts into a complicated product [Bon89]. The real problem seems to be how to get information out of the drawing and how to structure it. An answer to this difficult question is sought in a separate project. In the work described in this thesis it was assumed that a detailed description of the product is available (simulated by hand input).

8.4 Testbed systems

Parallel to the development of the Planner concept and the software prototype, some physical test sites were developed. After a thorough evaluation of candidate test-beds, two sites in the nearby Robotlab were selected [Hee88b]. Two sites were selected instead of one to preclude the development of a dedicated solution.

Testbed development started with the design and purchase of tools to support basic assembly tasks. To evaluate hardware performance, stand-alone programs were written in the systems own languages. In the last phase, assembly programs generated by the prototype Planner were tested on the two sites.

8.4.1 The ASEA test site

Central element in the ASEA site is an ASEA IRB-1000 robot. The robot has six degrees of freedom and was specially designed for small product assembly. It has a pendulum type of construction that allows rapid acceleration and can handle a up to 3 kg. The robot is mounted on a gantry, which is controlled as an external axis from the robot controller (fig. 8.4).

The gantry provides an opportunity to work on several workplaces, but also effectively enlarges the working area within one workplace. The robot was originally equipped with an Asea multigripper system, that could support six to eight grippers. Much of the space on this multigripper was taken

Figure 8.4: *Layout of the ASEA test site*

for training and demonstration purposes. Therefore, a new, exchangeable multigripper system was constructed, and the robot was equipped with a standard Tool Exchange System (manufactured by Reinoldus).

The wrist of the robot can rotate two full revolutions. To prevent air hoses from getting tangled around the wrist, the robot has to be equipped with a reel construction. The original ASEA reel construction accommodated twelve air connections, but had no electrical signals. A new reel was therefore constructed.

For assembly of the Cranfield Benchmark, the ASEA site supports two different tool change strategies: multigripper and exchange of single part grippers. The basis is a Tool Exchange System (TES). A TES mountable multigripper was equipped with three small, pneumatically actuated grippers. All three are of the open-or-close type with rotating fingers. New fingers were designed for the Benchmark parts. Steps in the fingers of one grippers make it suited to handle both the spacer block and the handle.

Alternatively, there is a set of three single grippers that can be mounted directly on the TES. Because the same TES was chosen for both the Asea site and the ROVIS site (see next section), the effectors can be used at both

sites. To support automatic tool changes a tool rack with three positions was constructed.

The site is equipped with a flexible pallet transport system (Philips VTS). In the experiments it was used to implement and test the sub-batch strategy. A extra pallet station was added to the original layout and set of new part and product carriers was designed.

The two testbed systems have a different control architecture. The emphasis of the tests at the ASEA site was on the influence of parts feeding, tool changing and product transport strategies (see chapter 3). For the control concept, a "traditional" approach was chosen: The robot controller is used as workstation controller (see fig. 8.5). The process plans generated by the Planner are transformed into sequences of commands in the robot's off-line programming language. A PC-based compiler, developed by the Robotlab staff, translates the off-line program into a robot program. Synchronization between the assembly workstation and the pallet transport system is simulated by a human operator.

8.4.2 The ROVIS site

The second site that was used for experiments is called Robot VISION (ROVIS) site. Central element in the site is an IBM-7545 SCARA robot (fig. 8.6). The upper arm has a length of 400 mm and a range of 200 degrees. The lower arm has a length of 250 mm and a range of 180 degrees.

On the arm, tools can be mounted via a standard tool exchange system (Reinoldus). The exchange system supports six pneumatic channels and 18 electrical contacts. Two of the electrical contacts are used for tool identification. The exchange system is actuated pneumatically.

During early tests, we found that the pneumatic and electrical cables tended to tangle around the z-axis of the robot when the wrist was rotated. Sometimes they got stuck and a servo-error was generated. A wire guide and reel construction was developed to keep the 7 pneumatic and 18 electrical connections in a neat spiral shape.

Three pneumatically actuated grippers are available. The grippers are of the open-or-close type, two with rotating fingers, the third has sliding fingers with a 20 mm stroke and can be used for multiple parts. When not in use, the grippers are mounted in a tool rack. Sensors in the rack guard tool

TCI = Transport Controller Interface

TC = Transport Controller

RC = Robot Controller

VTS = Variable Transport System

TES = Tool Exchange System

Figure 8.5: Control Architecture of the ASEA site

Figure 8.6: *Layout of the RObot VISION (ROVIS) test site*

availability. The grippers can be automatically exchanged.

A simple commercial vision system (a Honeywell HVS-270 line-scan camera) is used as a fourth end-of-arm tool. Through a standard interface plate, the vision system can be exchanged and mounted on the robot arm in the same way as a gripper. The system consists of a small box that contains both a camera and the image processing hardware. The supply of electrical power and the communication and synchronization signals all run via the electrical contacts of the tool exchange system, so no extra wires are needed. The vision system has a resolution of 1×256 pixels (with 256 grey values each) and communicates via a serial line and some separate binary synchronization lines. It can be preprogrammed to detect or count edges, transmit the edge position, etc.

To test guarded motions, parts can be fed on a small transport belt. Near the belt there is a special device that can rotate locking pins over ninety degrees. The device ("flip") represents a typical engineering solution to the orientation problem that arises because the pins are fed vertically on the Cranfield Benchmark base plate, while they have to be inserted horizontally.

The emphasis of the tests in the ROVIS site was on lower level planning and control issues, therefore relatively more effort was put into the design of a well structured control system. A new control structure was designed,

Figure 8.7: Control structure of the ROVIS test site

modeled after the hierarchical control architecture (described in chapter 2). Figure 8.7 gives an overview. On the top level there is the microVAX, that runs the Planner software, and acts as cell/line level controller. In the second layer we find an IBM-PC/AT and an ACRO 900 I/O-box, that together act as the workstation level controller. The robot is controlled directly from the microVAX, where a new robot driver was created based on Calvin, a robot control package that came available through another project [Mar86], [Mar87].

The robot interface has a simple structure. Table 8.3 gives an overview of the commands that are used. The only program that is resident in the robot controller is a small "motion interpreter" written in the original robot language AML/Entry.

There were two reasons to maintain a direct link between the Vax and the robot controller: The Vax is better equipped to handle asynchronous I/O and the database containing the motion descriptions runs on the same machine.

Robintf Command	Purpose	Use
INIT	Home, Load motion interpreter program and initialize program	Is automatically called on first move command
POSITION	Ask robot for current position	To get robot position after a guarded motion
AMOVE	Move absolute to position in world coordinates	Unguarded move
GMOVE	Guarded move. Specifies port number and stop condition. Sets guard, calls AMOVE and resets guard.	Guarded move

Table 8.3: *Robot interface in distributed architecture*

8.5 Batch level experiments

Experiments at the batch level were executed to examine the influence of gripper exchange strategies, parts feeding strategies and the sub-batch strategy on the assembly cycle time. Table 8.4 provides an overview of the main experiments. All experiments were done with the nine part version of the Cranfield Benchmark in the Asea site. For kit assembly the benchmark base plate was mounted on a pallet that was stationed in the work area of the robot. For the experiments with the sub-batch strategy, a set of six benchmark products was created. All experiments were executed at 20% and 100% of the robot speed.

Results were gathered in the form of timing tables. Table 8.5 summarizes the results of the batch level experiments. A more detailed description of the experiments can be found in appendix C.

8.5.1 Gripper exchange strategies

Three gripper exchange strategies were evaluated: Multigripper, Gripper Exchange System, and Multi-feature gripper.

As expected, the multigripper strategy works faster than the gripper exchange strategy. Gripper exchanges take up more than 50% of the total time of TES (Tool Exchange System) based assembly. The average gripper exchange takes three times the assembly time of a single part. This is more

Exp.	Product	Tools	Feeding	Transport	Speed
1.	CB9	MG	Kit	-	20%
2.	CB9	MG	Kit	-	100%
3.	CB9	TES	Kit	-	20%
4.	CB9	TES	Kit	-	100%
5.	6*CB9	TES	Pallet	Sub-batch	20%
6.	6*CB9	TES	Pallet	Sub-batch	100%
Site	Asea				
Date	December 1989 / January 1990				
CB9	Nine part version of the Cranfield Benchmark				
MG	Multigripper				
TES	Tool exchange system				

Table 8.4: Overview of batch level experiments

Exp. No.	Total Assy Time	Tool Change Time	Rack to Pallet Motion	Part Pallet Change Time	Product Pallet Change Time	Net Assy Time
1	77	-	-	-	-	77
2	48	-	-	-	-	48
3	181	99	20	-	-	62
4	104	52	12	-	-	40
5	975	117	15	162	122	663
6	725	53	12	148	123	442
Average time per product						
5	163	20	3	27	20	111
6	121	9	2	25	21	74

Table 8.5: Timing results of the batch level experiments

than expected. Two explanations are offered: The tool rack is mounted in a rather awkward position on the side of the VTS (the robot has to use the external axis, and many intermediate points). Also, the net assembly time (without tool changes) is biased because the parts are fed in a kit, resulting in very short feeder-to-product motions.

After correction for tool exchanges and the subsequent tool-rack-to-kit motions, the assembly cycle time with TES mounted grippers is comparable to multigripper assembly (actually TES based assembly it is a little faster, because the construction of the multigripper sometimes causes the robot to make more complex motions)

The multi-feature gripper (e.g. TES gripper no. 2, which handles spacer blocks, shafts and handles) seems to offer the best of both worlds: It saves gripper exchange time and its construction keeps motions simple. The only problem is that multi-feature grippers can handle a limited number of parts types. Development of a more universally usable gripper therefore still appears worth while.

8.5.2 Sub-batch assembly with TES mounted grippers

Figure 8.8 gives an overview of the physical layout of the Asea site. The pallet transport system supports two "pallet loops", one for part pallets and one for product pallet. Both loops circulate through a general pallet buffer area.

As expected, sub-batch assembly reduces the effect of gripper exchange time on the total assembly cycle time. On the other hand, a new 30% overhead is introduced, in the form of product and part pallet exchanges. Careful design of the transport system layout can reduce this overhead (see section 8.5.4).

8.5.3 Feeding strategies

Comparing the *net* assembly times of TES-based sub-batch (pallet feeding) assembly and TES-based kit-feeding assembly, shows that pallet feeding is significantly slower. The main cause is the larger distance between the feeding and mounting positions. Treating assembly and parts feeding as separate problems, one could conclude that kit feeding is better. In most practical cases however, one has to weigh in the extra cost of putting the parts in the kit. Also, systems with fixed feeders usually have even larger distances in

Figure 8.8: Pallet Flow in the ASEA site using the sub-batch principle

between feeding and mounting positions. Therefore, for assembly systems requiring a high adaptability and planning flexibility, sub-batch assembly still seems a promising concept.

8.5.4 Other system design influences

Robot speed

To examine the influence of the speed of the robot on the assembly cycle time, all experiments were run at 20% and 100% of the maximum robot speed. Analysis of the timing results in table 8.5, clearly shows that an increase in the speed of the robot leads to faster assembly. The effect is not, as one might expect, an 80% reduction of the cycle time, but a reduction of about 40%. The main reasons are that every pick-and-place cycle includes several actions that do not depend on robot speed (e.g. built-in delays to let a gripper fully close or open), and that for short transfer distances not robot speed but acceleration is the critical factor.

Layout of the pallet transport system

To examine the influence of potential improvements in the layout of the pallet transport system some experiments with the VTS were executed. Table 8.6 gives an overview. Comparison of the results in this table with the results of the sub-batch experiments described in the previous section, leads to the conclusion that the layout of the pallet transport system has a significant impact on the assembly cycle time. Potential improvements can e.g. come from replacing manual synchronization with automatic synchronization and from placing VTS at better position with respect to the robot (eliminates the need to use the external axis).

Some other observations:

- The relatively fast product pallet exchanges near the end of the second sub-batch experiment (see table C.13) were obtained by anticipating the arrival of the product pallet. This strategy can be further exploited to minimize product pallet exchange overhead.
- Whereas obtaining a new product pallet from the product buffer took about 8 seconds (experiment VTS-2), obtaining a new part pallet from

Experiment	From	To	Via	Time	Remarks
VTS-1	3	3	4	74 s	6 pallets in pallet buffer
VTS-2	3a	3	-	8 s	Buffer around the corner
VTS-3	1a	1	-	5.7 s	In-line buffer
VTS-4	4	5	-	33 s	
VTS-5	5	5	4	53 s	3 pallets in pallet buffer
Site	Asea / VTS				
Date	12 December 1989				
Remark	Times are average of at least three repetitions				
Note	For station names see station table				

Table 8.6: *Pallet transport times*

the pallet buffer area took about 33 seconds (experiment VTS-4). Introduction of a buffer for part pallets can lead to a significant reduction of part pallet exchange overhead.

- Comparison of a pallet exchange in a station with an in-line buffer (station 1, visible on the left in fig. 8.4) with the product pallet buffer "around the corner" used in the sub-batch experiments, shows that some improvement in product pallet exchange time can come from a different product buffer configuration (5.7 s vs. 8 s, see experiments VTS-4 and VTS-2). This improvement can be significant because product pallet exchanges occur frequently.
- The whole "train" of product pallets does not need to cycle through the product position: the last product pallet of the train can remain in the product position to become the first pallet of the next train.

8.6 Product level experiments

As described in chapter 5, the final implementation of the sequence planning algorithms was based on a combination of a disassembly approach (remove parts one by one, the reverse of this sequence is a valid assembly sequence) and a relation based approach (establishing precedence relations by examining the type of connection between parts). The validity of this approach was first checked manually [Guu89], later experiments were executed with a

Product	User Defined Subassemblies	EE	Input	CPU time (s)	Nodes in LASTD
CB9	-	9	File	6 ¹	31
CB17	Locking pins	10	File	7 ¹	33
CB17	Locking pins	10	Database	12 ¹	33
DIAC	Supports, Vertical slider	11	File	12 ²	N.A.
DIAC	Supports	17	File	31 ²	N.A.
DIAC	-	21	File	30	200
EE Effective number of elements (UDFS effectively treated as a single part)) ¹ Time includes stability check and expansion) ² Time includes stability check					

Table 8.7: *Sequence plan computation times for various products*

prototype sequence Planner, written in C [Hee89]. Products that were used include the Cranfield Benchmark (in a 9 part and 17 part version), and a test product designed for the DIAC project (21 parts). Table 8.7 provides an overview of sequence plan computation times for various products.

8.6.1 The Cranfield Benchmark

Clusters and stability

The clustering mechanism was shown to work effectively. For the 9 part version of the Cranfield Benchmark (no locking pins), clustering reduced the number of sequences from a theoretical $9! = 362880$ to an LASTD describing a mere 128 (partial) sequences. Application of the three basic stability rules further reduced this number to 37. The LASTD in fig. 8.9 was manually generated from the actual sequence Planner output.

The prototype cluster generation procedure for the benchmark product proceeded very much like the the description provided in section 5.5. An extra routine to detect and eliminate redundant clusters was added: Symmetry may cause one group of parts to be detected as two different clusters when two different parts can be used as first removable part. One of the two

Figure 8.9: Total LASTD for the nine part version of the Cranfield Benchmark.

clusters can be deleted without losing information.

The data structures in the Planner implementation are somewhat more complicated than the graph-like relation networks and LASTDs presented in the example, because they contain "extra" information for bookkeeping during planning. Also, for every disassembly step the prototype implementation creates a new level in the computer representation of the LASTD. In other words, what is represented as a single top level in fig. 8.9 is in fact a concatenation of the top two levels of the LASTD in the computer. The top level in the computer LASTD only indicates that side plate 2 should be assembled after the rest of the parts (the first step in disassembly is the last step in assembly). Level 2 of the computer LASTD indicates that inside this rest, side plate 1 has to be assembled before the layer cluster. Fig. 8.9 represents the same information in a slightly more compact form.

Automatic generation of the LASTD for the Cranfield Benchmark from the part list, the relation network and the part position network took about 6s CPU time on a MicroVax II, using a file based product description. The code was not optimized, improvements in execution speeds are possible.

Sequence expansion

As a last step in sequence planning, the LASTD is further reduced by an expansion mechanism, that interacts with lower level Planners a.o. to evaluate the applicability of grippers. Proper operation of expansion and optimization was verified manually, using gripper data from the physical test beds. In the two test sites, there are two different sets of tools available. Table 8.8 gives an overview of the parts that are handled by each of the tools.

In each of the two sets there is one gripper specialized in handling side plates. The peg gripper on the multigripper can also be used for shafts, exploiting the fact that the body of the peg is cylindrical and has the same diameter as the end of the shaft. The fingers of the third gripper on the multigripper have a step such that the gripper can handle both the handle and the block: a typical example of a multi-feature gripper.

The peg gripper in the TES set handles pegs by their (more or less) cylindrical end, which has the same diameter as the locking pins. The third gripper in the TES can handle the shaft, the handle and the block, largely because the gripper has a large stroke and parallel moving fingers.

Part	Multigripper number	TES gripper number
Side plates	1	1
Pegs	2	2
Locking pins	-	2
Shaft	2	3
Handle	3	3
Block	3	3

Table 8.8: Overview of gripper suitability for Cranfield Benchmark parts.

For sequence expansion, the interesting part of the LASTD in figure 8.9 is the sequence inside layer cluster LA_1 (at the top level there is only one fixed sequence, and inside sort clusters there are no tool changes). Assume that the TES based set of grippers is used. Assigning a symbolical cost factor of 1 for every tool change while stepping through the layer creates 32 different paths between start and end node. The cost of these paths (the number of tool changes) varies from 2 (e.g. sequence $S_1S_2B_lS_hH$) to 5 (e.g. sequence $S_hS_1B_lS_2H$). Figure 8.10 shows the four "cheapest" sequences in a tree like structure.

8.6.2 Other products tested

CB17

A second test of the sequence planning algorithm with was done with the full Cranfield Benchmark (17 parts, including the eight locking pins). Because each locking pin has a unique combination of relations (one relation with one of the four spacer pegs and a second relation with either the top or the bottom side plate), the locking pins are not detected as a sort cluster. The algorithm still works, but the result is an enormous increase in sequence alternatives, most of which are irrelevant.

In part, the artificial structure of the Benchmark product is responsible for this explosion of alternatives, and one would like to sidestep this complexity. On the other hand, in some situations one might be interested in such a detailed description of all sequence possibilities. The User Defined Sub-assembly provides a way to focus the attention of the sequence Planner on

Figure 8.10: The four "cheapest" sequences for assembling the nine part version of the Cranfield Benchmark with the TES based set of grippers.

the interesting parts in a product. For the 17 part Benchmark, e.g. one could create an artificial cluster ("subassembly") of the eight locking pins, effectively reducing the number of elements ("parts") to be dealt with to 10.

Automatic generation of the LASTD for the 17 part version of the Cranfield Benchmark with the locking pins in a user defined subassembly took about 7s CPU time on a MicroVax II, using a file based product description. Running the same test, using the database as input, took about 12s CPU time.

The DIAC specification product

Another product that was used to explore the power of the sequence Planner was the DIAC specification product [Jon88, ch. 1.5]. Again an artificial product, it was especially designed for the DIAC project, incorporating several new features not in the Cranfield Benchmark:

- A larger number of parts
- A larger variation in part shapes
- Multiple assembly directions
- Subassemblies

Figure 8.11: *DIAC specification product, mainly consisting of rectangular parts (Medusa 3D model)*

Figure 8.11 shows the product in assembled form. LASTD creation for the full 21 part product took about 30s CPU time. This test did not include

stability checking. The resulting network had 200 nodes. Creating three user defined subassemblies for the parts on the vertical "slider" and the two "supports" on this slider effectively reduced the number of elements to 11. Planning time reduced correspondingly to about 12 s CPU time.

8.7 Part level experiments

The part-script was implemented as a C-language data structure (linked list). The search algorithm was modeled after the product-level search algorithm used in sequence expansion. The cost factor for the estimation primitives was given a fixed value (independent of the part involved). The cost factor for the simulation primitives used in the prototype, was derived from the Boothroyd codes. For an overview see appendix D. The implementation provided no problems. Correct operation was verified manually.

The algorithms for position propagation were implemented according to the combined approach described in chapter 6. A model of the ROVIS site was used to evaluate the performance. Positioning accuracies for the robot, gripper exchange system etc. were estimated from robot supplier documentation and on-site measurements.

The implementation of the position propagation algorithms proved to be quite complicated. It was e.g. not clear exactly when to switch from worst case descriptions to gaussian distributions. The conservative approach turned out lead to large estimates for the uncertainties in the ROVIS environment. Even in simple cases the estimates of net uncertainties were unrealistically large. The largest cause of trouble appeared to be the interdependency between orientation and position. An example:

Imagine a situation where a robot with a gripper in its TES is ready to grasp a part from a feeder. The critical relation is the relation between the gripper and the part: The question is whether the position of the part is known well enough to make the grasp operation successful. The feeder is in a fixed position with respect to the robot, 1 meter from the robot base. It is assumed that the engineering staff has done its job quite well, and that the feeder is positioned within 0.2 mm of its intended position in the horizontal plane and an angular accuracy of 0.01 radians (less than one degree) around the vertical axis. Further it is assumed that the robot is a well-calibrated assembly robot with an absolute positioning accuracy of 0.02

Figure 8.12: *Uncertain relations in a simple assembly environment*

mm in the horizontal plane. The part is on the feeder, 20 mm above the feeder reference point.

The nominal position of the feeder is 1 meter from the robot:

$$\vec{x}_1 = (x, y, z, u, v, w) = (0, 1000, 0, 0, 0, 0) \quad (8.1)$$

The uncertainty vector for the feeder is

$$\hat{x}_1 = (d_x, d_y, d_z, d_u, d_v, d_w) = (0.2, 0.2, 0, 0, 0, 0.01) \quad (8.2)$$

The nominal position of the part is 20 mm above the feeder:

$$\vec{x}_2 = (0, 0, 20, 0, 0, 0) \quad (8.3)$$

The uncertainty vector for the part is:

$$\hat{x}_2 = (0, 0, 0, 0, 0, 0) \quad (8.4)$$

At the moment the gripper is commanded to grasp the part its position of with respect to the robot is:

$$\vec{x}_3 = (0, 1000, 20, 0, 0, 0) \quad (8.5)$$

The uncertainty vector for the gripper is:

$$\hat{x}_3 = (0.02, 0.02, 0, 0, 0, 0) \quad (8.6)$$

To compute the net relation between part and gripper, first the relation between the part and the feeder is inverted. The resulting nominal relation between feeder and part:

$$\vec{x}_4 = \vec{x}_2^{-1} = (0, 0, -20, 0, 0, 0) \quad (8.7)$$

And the associated uncertainty vector:

$$\hat{x}_4 = (0, 0, 0, 0, 0, 0) \quad (8.8)$$

Also the relation between robot and feeder must be inverted:

$$\vec{x}_5 = \vec{x}_1^{-1} = (0, -1000, 0, 0, 0, 0) \quad (8.9)$$

The uncertainty vector clearly expresses the interdependency between orientation and position: the small angular misalignment in the orientation of the feeder is multiplied by the large distance between feeder and robot:

$$\hat{x}_5 = (10.2, 0.2, 0, 0, 0, 0.01) \quad (8.10)$$

Compounding of x_4 , x_5 and x_3 results in the critical relation:

$$\vec{x}_6 = (0, 0, 0, 0, 0, 0) \quad (8.11)$$

implying that the nominal position of the part is the same as that of the gripper. The computed uncertainty however, is large:

$$\hat{x}_6 = (20.22, 0.22, 0, 0, 0, 0.01) \quad (8.12)$$

In x-direction (a conservative approximation) the uncertainty is more than 20mm! This is well beyond the grasp capabilities of many common grippers.

The question arises why such a large uncertainty can occur, while a relatively accurate environment was assumed? Another question is why on-line (teach-in) generated programs usually do work in less calibrated environments?

The answer to both questions partially lies in the way of modelling, in the conservativeness of the approach and in the way the model is used. In this simplified example, computing the net uncertainty the other way around (from the gripper to the part) leads to a remarkably different result: An estimated net uncertainty of 0.22mm in both x and y directions! The reason is that no orientation uncertainty was modeled in the gripper position. This, unfortunately, is not realistic. Flexible assembly robots will be equipped with a TES, always causing some orientation misalignment. The conclusion was that the way of modeling needed improvement.

The difference with on-line teaching is also significant. The key is that on-line teaching implies on-line calibration. The operator directly controls the critical relation while programming. Gripper, robot, feeder etc. may all be somewhat "misaligned" (the word effectively has little meaning in a teach-in environment), as long as the relative positions of the objects during programming are maintained until program execution time, the high repeatability of the robot will faithfully reproduce the taught positions.

To obtain realistic results with the experiments, a whole set of measures was taken. A new model for robot motions was introduced: the robot is now assumed to achieve positions with absolute accuracy with respect to every fixed position inside the assembly working area (e.g. the base of the feeder in the example above). Validity of this model was ensured by a series of calibrations.

The robot controller internal model was updated with a manufacturer supplied calibration kit. Calibration data suggested that in the internal representation a.o. arm lengths were adjusted with several tenths of millimeters. The fixed part of the TES was mechanically aligned on the robot, the grippers were aligned on the TES and the fingers were aligned on the grippers. Tool TCP offsets were measured and stored in the world model database. This last chain of alignments can be compared with adjusting and calibrating the of tool tip offset for NC machining tools.

Several reference points were established in the assembly environment (e.g. by putting location markers on the tool rack). The "real" positions of these reference points (as far as the robot is concerned) were measured by on-line calibration (good old teach-in) and stored in the world model database.

To simplify prototype testing, only a subset of the position propagation algorithms was used. All uncertain relations were described with gaussian distributions. The final result was that the Cranfield Benchmark could be

successfully assembled.

Another point that became clear during the experiments, was the importance of the self aligning capability of many grippers. The position of a part may not be known accurately before it is grasped, but as long as the gripper fingers go around it and have the right geometry, the gripper may still get a good hold. Either the part slips in the feeder or the relative position of the gripper aligns itself to the part by compliance in the gripper or robot. The result of a successful grasp operation is that the position of the part in the gripper is known with more certainty: An example of mechanical reduction of uncertainty. The effect was incorporated in the position propagation software: The computed net uncertainty between gripper and part is replaced by a fixed, gripper dependent uncertainty if simulation indicates that a successful grasp is guaranteed.

8.8 Primitive level experiments

General operation of the primitive level Planner was demonstrated by successful execution of Planner generated programs for the Cranfield Benchmark. Detailed operation all primitives was evaluated by hand. Three examples of experiments that were executed are presented in this section: grip selection, feeding a part on a belt, and the introduction of a new piece of hardware.

8.8.1 Select_grip

To verify correct operation of the grip Planner, the set of available grippers was enlarged. All Planner generated grips for all parts of the Benchmark were systematically evaluated by hand. This revealed some interesting aspects of the influence of feeder design on grip selection. Also, the Planner found some "good" grips that were overlooked by the experienced programmers in earlier experiments. Some examples:

- To locate the spacer block and the side plates on the benchmark base plate, there are locating pins in the same holes as used later for mounting. This leaves much room for grasping. An interesting grip that tended to be overlooked is e.g. the lengthwise grip of the spacer block.

- Spacer pegs can be grasped on the thick middle flange. This is more stable because fingers are further apart and the grip center is closer to the center of gravity of the peg. This "discovery" was used in the design of the second gripper for the multigripper in the Asea site.

8.8.2 Part feed on belt

The preparation of scheduler and dispatcher actions was evaluated by examining the procedures around feeding one of the side plates on a small transport belt in the ROVIS site. First, the belt is set into motion. A light gate monitors the presence of a part near the end of the belt. As soon as it detects a part, the belt is signalled to stop (belt motion was implemented as a guarded action). The robot moves in, with the line-scan camera mounted in its TES. Three line scans under an appropriate angle with the part deliver enough information to compute the exact position of the part. Because the field of view of the camera is relatively small, the mobility of the robot is used to enlarge it. For this, both devices are coordinated in a guarded motion, as described in chapter 7.

Operation of the guarded action and the three point measurement procedure were evaluated. The line scan camera was mechanically aligned with the TES. The offsets of the scan line and the pixel to millimeter conversion factor were measured. Environment light turned out to be a critical factor. In its basic mode, the camera automatically adjusts the integration time in such a way that the whole image has a certain average gray level (similar to a largest contrast). In the application, the robot moved the camera from a dark area (the belt surface) to a light area. In the dark area the camera would assume a very long integration time. Together with the motion enforced by the robot, this "blurred" the edge of the side plate so badly that it was not recognized. A solution was found in an increased light level and fixing a "fast" exposure time in the camera. The final result was that the side plate could be successfully grasped from the belt.

The three point measurement procedure operated as expected. The computed part position was accurate enough to make the following grasp operation successful. As expected, having the camera in the TES made the total operation rather slow (several minutes). In a separate project (using the same robot) it was shown that operation times in the seconds range are feasible with an arm mounted (instead of TES mounted) 2D camera.

8.9 Evaluation

8.9.1 Planning

Batch level

Application of the sub-batch strategy reduces the influence of the gripper exchange time on the overall assembly cycle time. On the other hand, new actions are introduced that can slow down the process: Product pallet exchanges and part pallet exchanges. Careful design of the transport system layout (including e.g. the introduction of buffer stations in strategic places) can keep the overhead of pallet exchanges low.

In the experiments with the sub-batch strategy, the same pallet transport system was used for parts and products. This is an improvement over the IVF-KTH Mark II concept, because putting back actions of finished products are eliminated. On the other hand, it puts extra constraints on the transport system layout.

With a multigripper, the product pallet exchange time could be further filled with acquiring new parts from the part pallet: A strategy named Multigripper Ride Sharing. While the product pallet is exchanging, the robot acquires not just one, but several new parts from the part pallet, making small movements and indexing the multigripper in between. Then the robot moves over to the product pallet and inserts the parts one by one, again making small movements and indexing the multigripper in between. The parts effectively "share the ride" from part pallet to product pallet.

Another improvement would be a transport system with smaller sized pallets: It would lead to a smaller distance between feeding and assembly positions. Also, the external axis of the robot would not have to be used.

Product level

The experiments showed that, when an accurate description of the product is available, automatic generation of feasible sequences is possible, in a limited amount of time (CPU time in the seconds range). Verification was done by manually drawing a graphical form of the LASTD. This was easy for the Cranfield Benchmark, but for the 21 part version of the DIAC product (200 nodes in the LASTD) it takes a long time. Automatic generation of

the graphical form could be of help. For manual evaluation, assigning some parts to "user defined subassemblies" provided powerful support by helping to "focus" the attention of the sequence Planner.

The first part of assembly sequence planning can be executed independent of the execution environment and because the information needed for this phase (the product design) is available on a longer time scale. During this phase computation time constraints are less important.

Cluster recognition algorithms were embedded in a modular structure. Several of these recognition "experts" could be run in parallel. Another advantage of the structure is that it makes addition of new "experts" relatively easy.

Manual introduction of sequence constraints (e.g. "part *A* should always be inserted before part *B*" or "subassembly *XYZ* is not stable and should be eliminated") could further improve the practical applicability of the sequence Planner.

The sequence Planner was demonstrated to work automatically for "artificial" products. In a manual evaluation of the planning algorithms, several "real world" products were used (gas valve, head of an electric shaver, pneumatic valve, telephone). In these products some features were encountered that suggested ways to enhance the sequence Planner:

- Insertion motions other than a straight line
- Parallel assembly action on multiple parts (e.g. concurrent assembly of razor blades in a shaving head)
- Assembly of flexible parts (springs, membranes)

Several experiments with stability Planner prototypes have been executed [BvdDBT88], [BD88a], [BD88b], [Vri88]. Stability planning in a real assembly environment proved difficult because of the many degrees of freedom, the influence of friction and the interference of disturbing forces. Tolerances on part dimensions make it difficult to predict if and where contact will occur and finally, relatively simple geometric features of e.g. gripper fingers or fixture edges can have large consequences for part stability.

Stability was found a less important factor for assembly sequence planning, because it is hard to derive "strong" decisions from it. Tools and fixtures

play an important role. An example is the situation of two parts that do not touch each other. A "subassembly" of these two parts definitely has a stability problem. However, with a specially designed gripper there needs to be no problem.

Part level

Experiments at the part level demonstrated that automatic propagation of position accuracies in practical assembly environments is feasible. However, calibration and mechanical alignment play a very important role.

The part-script provided a useful basis for selection of device specific primitive actions. The choice of C-language for describing the part-script and creating the script search algorithm provided a large flexibility. However, it also makes the part-script less suited for end-user (system operator) modification. Alternative ways of describing the precedence relations between primitives could be explored:

Library and interpreter approach A system programmer creates a library-like database of standard (device-specific) primitives. Through a window based interface the end-user can select and sequence primitives into a interpreter based program. (ADEPT approach, [S*87]).

Network approach Assign a cost factor and cost estimate to all primitives (including planning primitives) and make precedence constraints between primitives explicit (e.g. "Select a gripper *before* grasping a part"). The part-script obtains a network structure similar to the ASTD of the product level and can be generated automatically from the precedence constraints. The end-user only manipulates primitives in terms of costs and constraints. Precedence between planning primitives of equal importance (e.g. select the robot or select the tools first?) can be influenced through planning cost estimates.

The prototype implementation used relatively simple estimation and "simulation" procedures to obtain the cost of primitive level actions. Future versions of the part level Planner could be interfaced to software packages that can accurately simulate primitive level actions (e.g. the GRASP package was shown to produce accurate predictions for robot motion time [Goo89]). In such an implementation the current "simulations" could be used as improved estimates.

The interdependency between position and orientation presented a problem in modeling series of relations. Small angular uncertainties are amplified when their effect is felt over a larger distance. A potential solution is proposed by Taylor [TR88]: to describe all dependencies explicitly in the form of linear equations. With this approach, the uncertainty area of fig. 6.8 could e.g. be exactly described with four equations of the type $ax + by \leq c$. The approach still suffers from the inherent conservativeness of the worst case approximation of the uncertainty distribution.

In some cases both planning and action primitives compete for the same resource: time. If the Planner runs in an environment where planning time is a critical factor (e.g. during on-line error recovery), the "cost" of planning itself (computation time) might be introduced in the evaluation function. Planning primitives would be assigned cost estimates and competing planning algorithms to perform the same planning actions at different cost and performance levels could be introduced.

In the first prototype implementation of the ROVIS workstation control software, the database with motions was queried separately for every motion. This approach provided beneficial flexibility during development (positions could be changed while the dispatcher program was active and the assembly robot acting as motion interpreter was "on-line"), but had a negative effect on the response time: It took about 25 s to generate a new motion on a lightly loaded Vax. The database queries were responsible for a large part of this delay. A significant improvement of response time could be obtained by using directly the motion parameters generated during planning.

Primitive level

Writing all primitives as C-procedures provided a large flexibility. Using an interpreter based language may further improve the open character of the software.

Robustness of plans

Robust plans contain alternative strategies for critical operations. The most flexible solution is to postpone all planning decisions that depend on the circumstances of execution until execution time (do all system dependent planning on-line). In the prototype all process planning was executed off-

line.

The search processes that operate at the various planning levels of the prototype Planner, are programmed to find only one (the most optimal) sequence of operations. To create more robust plans one can easily continue searching to find less optimal alternatives.

8.9.2 Testbed engineering

In an early stage of the project it was decided to verify the with a real assembly environment. This integrated approach certainly has had an extra value. People were more involved in the project, and it has had a higher "visibility" than similar projects relying on simulation.

On the other hand, real testbeds come at a higher cost in man-hours and material. Engineering was more complicated and took more time than expected. Some delays were caused by small problems, typical for this kind of systems: Non matching voltage levels, noise, inverted logic, odd implementations of "standards" (e.g. wiring of RS-232 connections). A serious source of delay was the limited availability of testbed sites.

One example of a more complicated problem: In the coordinate system of the IBM 7545 robot the directions of the X,Y and Z axes are defined as usual, according to an orthogonal, right-handed coordinate system. The rotation however, is declared positive as clockwise around the -Z axis. This "feature" remains unnoticed when a teach-in or hybrid programming method is used, because all positions are taught by hand. When the robot is programmed off-line, the robot positions are computed from a world model based standard right handed coordinate systems. The problem was solved by inverting the Roll-coordinate in the robot driver.

Other hardware developments

Several experiment designs were aimed at creating more flexible hardware for assembly systems: E.g. a new, two fingered gripper with parallel finger motion, a large stroke, and a high closing force [Sit87]. Another example is the passive arm, developed to attain the advantages of a second arm without the large investment of a full robot. The basic idea is to design an arm that has a construction similar to a robot, but has only brakes as actuators [Que87]. Due to time constraints, these hardware developments

never exceeded the prototype stage, but they did have a positive influence on the development of the Planner.

Chapter 9

Conclusions and recommendations

9.1 Assembly Process Planning

A hierarchical reference model for assembly process planning was proposed, in which the assembly process is decomposed into four levels of abstraction. The highest level covers batch oriented assembly tasks that are decomposed via product and part level tasks into primitive level actions that are tightly coupled to physical devices. The reference model proved to be an effective basis for exploration of assembly process planning problems.

Generating process plans for assembly is a complicated task at all abstraction levels, but the main source of complexity differs per level. The complexity of sequence planning at the product level is mainly influenced by the combinatorial explosion of sequence alternatives when the number of parts grows. At the part level, uncertainties in the positions and dimensions of objects dictate planning complexity. At the primitive level, reasoning on geometric details involves an explosion of information.

Starting from the four level reference model, this thesis describes a new, flexible approach for assembly process planning. Alternative strategies are created and evaluated at each level of abstraction. The optimization of assembly plans was focussed on assembly cycle time. To demonstrate the concept's feasibility, and to show that the chosen approach offers a generalized solution, a prototype planning system was built. This system (DAPPER,

an acronym for "Delft Assembly Process PlannER") successfully generated process plans for different products on different assembly systems.

Early experiments showed that not all input data required by the planner was available in a form that could be easily used. In the work described in this thesis an assumption was made on the availability of a detailed and explicit product description (simulated by hand input). It was shown that under this precondition, a planner can work.

At the batch level, several assembly strategies were examined. Apart from gripper exchange strategy, the speed and layout of the pallet transport system and the parts feeding strategy were shown to be important factors for cycle time optimization.

The sub-batch strategy was shown to reduce the influence of gripper exchange time on the overall assembly cycle time. On the other hand, new actions are introduced that can slow down the process: Product pallet exchanges and part pallet exchanges. Careful design of the transport system layout can keep this overhead low.

At the product level, rough sequence plans are created directly from the product description. To reduce the complexity caused by the combinatorial explosion of sequence alternatives, a new technique for grouping parts (clustering) and a new method for representing assembly sequences (the Layered Assembly State Transition Diagram) were introduced. The combination of these two techniques led to a significant reduction of the number of alternatives to be considered. Experiments showed that, automatic generation of feasible sequences is possible, in a limited amount of time (CPU time in the seconds range).

In the design of the prototype sequence planner emphasis was put on a modular structure. This structure makes it easier to extend the prototype with new cluster and relation types, new elimination heuristics and more detailed assembly cost estimates.

In the prototype implementation, stability of subassemblies was dealt with by simple heuristics using a simplified stability model. Several attempts were made to create a true stability planner. Experiments showed that a.o. friction and tolerances make this a more complex problem than expected. Stability was found a less important criterion for sequence planning than accessibility; the dependency of stability on small features of devices (grippers, fixtures) in the assembly environment makes it difficult to generate

powerful decisions.

At the part level, position uncertainties are propagated between primitive actions to verify that net uncertainties in critical relations remain within pre-defined error bounds. When necessary, (position) sensor actions are automatically included. A part script describing sequence relations between primitives, guides the search process. Experiments demonstrated that automatic propagation of position accuracies in practical assembly environments is feasible and that the part-script approach results in useful assembly plans. However, calibration and mechanical alignment play a very important role.

At the primitive level prototypes of many selection and action primitives were implemented. Most selection primitives were implemented in a rudimentary form, but grip planning was explored in more detail. For several parts of the benchmark product, grips were found that had not been identified by experienced robot programmers in early experiments.

The search processes that operate at the various planning levels of the prototype planner, were programmed to find only one (the most optimal) sequence of operations. To create more robust plans one can easily continue searching to find less optimal alternatives.

9.2 Testbed and prototype development

Two physical testbeds were built to demonstrate the execution of plans generated with the planner prototype. This integrated approach certainly has had an extra value. As a result, some parts of the planner obtained a different focus, and the influence of certain design decisions was again emphasized. People were more involved in the project, and it gave the project a higher "visibility" than similar projects relying on simulation. On the other hand, testbed building came at a high cost in man-hours and material. Engineering was more complicated and took more time than expected.

For one of the testbed sites, a new concept for cell control, based on intelligent, asynchronous devices was introduced. The robot controller was "degraded" to motion-interpreter. The result was a clear and open control structure.

Software development was supported by a PC-based Computer Aided Software Engineering (CASE) tool (PROMOD). The tool provided good support for consistency checks and the definition of a data dictionary. Automatically

generated dataflow diagrams provided an adequate means of communication in the development team.

The language C was chosen as basis for all the planner and cell control software. The main reason for this choice was standardization. C did provide a degree of flexibility that could not be offered by most existing robot programming languages. "Low level" software like robot-drivers and the link to the database were created with relative ease.

Descriptions of products and system were stored in a relational database system (INGRES). New requirements for describing products were formulated, including part-position- and part-relation networks. Resulting motions were also stored in the database.

9.3 Recommendations

An important function of the project was to cover flexible assembly from a broad perspective and to identify areas in which new or continued research could be rewarding. The following areas have been identified:

Product Model - CAD interface On the input side, the planner needs an accurate description of parts and products. State-of-the-art CAD systems do not contain enough information in an explicit form. Careful analysis of CAD models could make more of this information available.

Scheduling and dispatching On the output side, the planner assumes an intelligent and flexible assembly system to execute the plans its generates. The coordination of many asynchronous devices in the assembly environment is a complex problem. In the current implementation the planner does all optimization and generates linear plans. An intelligent scheduler could take over the final optimization of action sequences, based on the actual availability of resources.

Fine motion planning Grasping and mounting of parts becomes difficult when tolerances on object sizes or uncertainties in their (relative) position become so large that they can not be neglected anymore. A solution might be found in the combination of pre-defined strategies and intelligent control (compliant motion).

System Model - Interface to Layout planning As a counterpart to the work on the product model, a similar effort could be started that focuses on the system model: How to create a universal description for assembly tools, how to validate the system model against the real world?

Extension of the Batch Level The planning concept was restricted to a single product type and a single workstation. The concept could be extended to include variants of the basic product (creating a family level). Another extension of the concept is to allow more than one workstation (this will among others introduce new balancing and scheduling problems even for a single product type). Also, new cost factors could be introduced.

Further exploration of the Product level The current implementation can handle a limited number of relation types, corresponding to a (small) set of basic connections. More complex connections and new clustering heuristics could be explored. Concurrent assembly of multiple parts can be introduced. Although it is a less important factor than accessibility, subassembly stability has an influence on the selection of assembly sequences. General methods to assess stability of assemblies could be developed.

Alternatives for the Part level part-script The choice of the C-language as a basis for the part-script makes the latter less suited for end-user (system operator) modification. Alternative ways of describing the precedence relations between primitives could be explored: A library approach based on on a database of standard primitives, or a network approach, based on cost estimates and automated search.

New Primitives Introduction of more complex connection types is only possible when the actions are supported by more intelligent primitive operations. Also, new devices could be introduced in the assembly environment, requiring new primitive level actions.

Introduction of parallel processing In the current implementation all planning is done in a sequential fashion. The modular design of the planner makes it possible to run several parts in parallel.

Cell Control A more universally applicable workstation controller could be developed, that can coordinate multiple basically asynchronous, but intelligent and communicating devices.

Some of the topics described above have been identified as interesting in an early stage. The first three topics of the list above are now covered by separate projects in the larger context of the DIAC project.

The two testbed sites have been developed to demonstrate feasibility of the planning concept. The same systems could be used in the DIAC project to test new concepts (e.g. collision avoidance and grasp planning strategies), while the DIAC testbed is still under development.

Planner prototype tests have been executed successfully, but most experiments were based on artificial products. As a next step, further validation of the concept with industrial ("real-world") products is recommended.

Several companies that were involved in the project (through the user committee or student assignments) have shown interest in the ideas described in this thesis, indicating that further development of the planning concept is worth while. In the short run these companies could provide the industrial products needed for further validation of the concept. The companies would receive feedback in the form of comments on the product design.

9.4 The future

The experiments described in this thesis demonstrate that Computer Aided Process Planning for Flexible Assembly is feasible, and that it can support the generation of optimum assembly plans. On the other hand, many important decisions are made during product and system design. In part, process planning means dealing with the consequences of design decisions made earlier. To create truly flexible assembly systems that have a short cycle time, the results of process planning should be used as feedback to the designer. To achieve this, more integration between product design, system design and process planning is needed.

Development of process planning tools into more integrated systems is expected to take place along two lines: The first line is that of a tool to optimize system design for a given (family of) product(s). The work in this thesis biased towards this line. The second line is that of a tool to design products in such a way, that it can be optimally made on an existing system. The product design in such a system could be restricted by the operations available in the assembly system. The Concurrent Product and Process Design project at Stanford University is an example of this second approach.

Small adaptations in the system design can have important consequences in productivity. Therefore, this second type of system must include some features of the first. The "final" system will integrate both approaches.

In the short run, fully automatic process planning does not seem feasible. Thus, the route to integrated process planning must follow an approach that relies more on the interaction with a human planner and a gradual introduction automated reasoning tools. Both interactive and automated process planning would benefit from the development of better methods for providing cost estimates of part and primitive level actions.

Assembly systems are expected to evolve in a similar way NC machine tools have evolved the past decade. They will slowly shift from systems that were designed especially for one type of operation to systems that support certain "operation classes" (a term coined by Bakker [Bak89]), limited mainly by the tools that are available, the size of the working area and the number of axes of the assembly robot.

Bibliography

- [AA88] M.M. Andreasen and T. Ahm. *Flexible Assembly Systems*. IFS (Publications) Ltd., Bedford, UK and Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1988.
- [ABW84] E. Abele, R. Bäbler, and E.M. Wolf. Einsatzmöglichkeiten flexibel automatisierter Montagesystemen. *VDI-Z*, 126(13), July 1984.
- [AG87] A. Arnström and P. Gröndahl. A high speed small batch automatic assembly system with flexible feeding and automatic set-up. In *Proceedings of the 8th International Conference on Assembly Automation*, pages 237–248, IFS (Conferences) Ltd, March 1987.
- [AG88] A. Arnström and P. Gröndahl. Advantages of sub-batch principle in flexible automatic assembly as used in the IVF-KTH concept Mark II. In *Annals of the CIRP, Vol. - 37/1*, pages 9–12, 1988.
- [AKL83] M.M. Andreasen, S. Kaehler, and T. Lund. *Design for Assembly*. IFS (Publications) Ltd., Bedford, UK and Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1983.
- [All83] J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.
- [Baa88] J.P. Baartman. *Systematiek in het ontwerpen van montagevriendelijke producten*. Technical Report WPS-88.023, TU-Delft, Laboratorium WPS, May 1988.

- [Baa89] J.P. Baartman. *SMD Placement and The Estimation of Accumulating Spatial Uncertainties*. Master's thesis, TU-Delft, Laboratorium WPS, September 1989. Report no. WPS-89.033.
- [Bad85] M.A. Badger. A flexible automation system for diesel engine cylinder heads. In *Proceedings of the 6th International Conference on Assembly Automation*, pages 159-168, IFS Ltd, 1985.
- [Bak89] J.J.A. Bakker. *DFMS: Architecture and implementation of a distributed control system for FMS*. PhD thesis, Technische Universiteit Delft, June 1989.
- [Bas88] R. Bäßler. Rationalisation results in assembly-oriented design. *Assembly Automation*, 8(2):82-86, May 1988.
- [BD79] G. Boothroyd and P. Dewhurst. *Handbook Design for Assembly*. University of Massachusetts, Amherst MA, USA, 1979.
- [BD88a] N. Boneschanscher and J.H.M. van der Drift. *Automatic assembly sequence planning: the stability problem*. Master's thesis, TU-Delft, Laboratorium WPS, May 1988. First part, Report no. WPS-88.025.
- [BD88b] N. Boneschanscher and J.H.M. van der Drift. *Subassembly stability*. Master's thesis, TU-Delft, Laboratorium WPS, May 1988. Second part, Report no. WPS-88.024.
- [BDTG83] M. Bonney, M. Dooner, N.K. Taylor, and J.L. Green. Verifying robot programs for collision free tasks. In *Developments in Robotics*, pages 257-263, IFS, 1983.
- [BF81] A. Barr and E.A. Feigenbaum. *The Handbook of AI*. William Kaufman, Inc. Los Altos, CA, 1981.
- [BH89] N. Boneschanscher and C.J.M. Heemskerk. Grouping parts to reduce the complexity of assembly sequence planning. In *Proceedings of the 1989 IFAC/IFIP INCOM Conference, Madrid*, September 1989. Also: Technical report No. WPS-88.047, TU-Delft, Laboratory for Manufacturing Systems.

-
- [BHMS89] N. Boneschanscher, C.J.M. Heemskerk, P. Martens, and T. Storm. *Verslag Duitsland reis*. Technical Report WPS-89.017, TU-Delft, Laboratory for Manufacturing Systems, May 1989.
- [BJ83a] C. Blume and W. Jakob. Design of the structured robot language, srl. In *Proceedings of the conference on Advanced Software for Robotics*, 1983. Liege, Belgium.
- [BJ83b] C. Blume and W. Jakob. *Programmiersprachen für Industrieroboter*. Vogel-Buchverlag, Würzburg, 1983.
- [BN83] J. Boorsma and L.M. Nijdam. *Design For Assembly (in Dutch)*. VOA, Vereniging voor Bedrijfskunde, Neuhuyskade 40, 2596 XL Den Haag, 1983. Adapted from the "Handbook Design for Assembly" by G. Boothroyd and P. Dewhurst.
- [Bon88] N. Boneschanscher. *Grouping parts for the planning of assembly sequences, design and implementation*. Master's thesis, TU-Delft, Laboratorium WPS, May 1988. Third part, Report no. WPS-88.026.
- [Bon89] N. Boneschanscher. Notes wp2 meeting 9/10/89. October 1989. DIAC project.
- [Bou87a] J. Bouman. Automation and redesign: a recipe for successful assembly. *Assembly Automation*, 7(2):59-62, May 1987.
- [Bou87b] D.A. Bourne. *The Automated Craftsman Preliminary Research*. Technical Report CMU-RI-TR-87-22, Robotics Institute, Carnegie Mellon University, Pittsburgh PA, 1987.
- [BPM82] G. Boothroyd, C. Poli, and L.E. Murch. *Automatic Assembly*. Marcel Dekker, Inc., New York NY, USA, 1982.
- [Bro82] R.A. Brooks. Symbolic error analysis and robot planning. *The International Journal of Robotics Research*, 1(4), 1982. Also: MIT A.I. Memo No. 685.
- [BS82] S. Bonner and K.G. Shin. A comparative study of robot languages. *IEEE Computer*, December 1982.
- [BvdDBT88] N. Boneschanscher, J.H.M. van der Drift, S.J. Buckley, and R.H. Taylor. Subassembly stability. In *Proceedings of the*

1988 AAAI Conference, August 1988. St. Paul, Minnesota, USA.

- [BvdW87] M. Bijl and R. van der Waal. Praktikumhandleiding station 7: IBM SCARA robot assemblage. TU-Delft, Laboratorium voor Robotica en Flexibele Automatisering, 1987.
- [CIM88] Cimphony. Brochure from and private communication with Philips CFT, Eindhoven, 1988.
- [D*86] B.J. Davies et al. Artificial intelligence in manufacturing engineering. In *CIRP 36th General Assembly, Round table, Jerusalem - Haifa (Israel)*, 1986.
- [DH85] R. Dillman and M. Huck. Ein softwaresystem zur simulation von robotergestützten fertigungsprozessen. *Robotersysteme*, (1):87-98, 1985.
- [DIN67] DIN Norm 8593: Fertigungsverfahren Fügen. Beuth-Vertrieb, Berlin, 1967.
- [DL81] Y. Descotte and J.C. Latombe. GARI: a problem solver that plans how to machine mechanical parts. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 1981.
- [Dri88] J.H.M. van der Drift. *On the Propagation of Spatial Uncertainties between Assembly Primitives*. Master's thesis, TU-Delft, Laboratorium WPS, May 1988. Third part, Report no. WPS-88.020.
- [ESP86] ESPRIT. Project 384: integrated information processing for design, and control of assembly. 1986. Annual Report No.1 (1st April 1985 - 31th March 1986).
- [ESP88] ESPRIT. Project 384: integrated information processing for design, and control of assembly. 1988. Annual Report No.3 (1st April 1987 - 31th March 1988).
- [Fab85] F. Fabricius. An overview of flexible assembly automation. In *Proceedings of the 6th International Conference on Assembly Automation*, pages 33-40, IFS Ltd, 1985.

-
- [Fah74] S.E. Fahlman. A planning system for robot construction tasks. *Artificial Intelligence*, 5(14):1-49, 1974.
- [FK85] B.R. Fox and K.G. Kempf. A representation for opportunistic schedules. In *Proceedings Third International Symposium on Robotics Research*, Paris, October 1985.
- [Fox86] B.R. Fox. The implementation of opportunistic scheduling. In *Preprints of the International Conference on Intelligent Autonomous Systems*, pages 231-240, Amsterdam, December 1986.
- [Fox87] B.R. Fox. *A Representation for Serial Robotic Tasks*. PhD thesis, University of Missouri-Rolla, June 1987.
- [Gai81] A. Gairola. *Montagegerechtes Konstruieren: Ein Beitrag zur Konstruktionsmethodik*. PhD thesis, TH Darmstadt, 1981.
- [GEI87] GEI. *ProMod User's Manual*. GEI, Aachen, West-Germany, 1987. Last update: V1.6a, September 1988.
- [Goo87] R.H.M. Goossens. *Uitgevoerde montagesystemen*. Technical Report WPS-87.058, TU Delft, Laboratorium WPS, October 1987.
- [Goo89] R. Goossens. *Een evaluatie van het robot off-line programmeer- en simulatiesysteem GRASP*. Master's thesis, TU-Delft, Laboratorium WPS, May 1989. Report no. WPS-89.013.
- [Guu89] G. Guurink. *Een evaluatie van een clusterherkenningsprocedure*. Technical Report WPS-89.008, TU-Delft, Laboratorium WPS, March 1989.
- [Har85] J. Hartley. Large-scale assembly system for printers. *Assembly Automation*, 5(3):160-164, August 1985.
- [Har86a] J. Hartley. Robotic assembly of diesel injection components. *Assembly Automation*, 6(3):91-94, May 1986.
- [Har86b] J. Hartley. Yaskawa moves into the assembly market. *The Industrial Robot*, 38-41, March 1986.

- [Hay87] C. Hayes. *Using Goal Interactions to Guide Planning: The Program Model*. Technical Report CMU-RI-TR-87-10, Robotics Institute, Carnegie Mellon University, Pittsburgh PA, 1987.
- [HdMS86a] L.S. Homem de Mello and A.C. Sanderson. *AND/OR Graph Representation of Assembly Plans*. Technical Report no. CMU-RI-TR-86-8, Robotics Institute, Carnegie Mellon University, Pittsburgh PA, August 1986.
- [HdMS86b] L.S. Homem de Mello and A.C. Sanderson. And/or graph representation of assembly plans. In *AAAI-86: Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 1113-1119, Morgan Kaufman Publishers, 1986.
- [Hee86] C.J.M. Heemskerk. Voortgangsrapport project 44.0740 periode september 1985 - maart 1986. Spring 1986. Distributed at the STW user group meeting.
- [Hee87a] C.J.M. Heemskerk. Programming an intelligent assembly cell. In *Proceedings of the First European Symposium on Assembly Automation*, Vereniging voor Produktietechniek VVW, March 1987. Veldhoven, The Netherlands.
- [Hee87b] C.J.M. Heemskerk. Voortgangsrapport project 44.0740 periode maart 1986 - december 1986. Spring 1987. Distributed at the STW user group meeting.
- [Hee88a] C.J.M. Heemskerk. DIAC test ADEPT. July 1988. Intern rapport TU-Delft DIAC Project, wp1.
- [Hee88b] C.J.M. Heemskerk. Voortgangsrapport project 47.0740. May 1988. Distributed at the STW user group meeting.
- [Hee89] C.J.M. Heemskerk. *Plan Product: Design of an Assembly Sequence Planner*. Technical Report WPS-89.002, TU-Delft, Laboratory for Manufacturing Systems, January 1989.
- [HL88a] I. Ham and S.C.Y. Lu. Computer-Aided Process Planning: the present and the future. In *Annals of the CIRP, Vol. - 37/2*, pages 591-601, 1988. Keynote Paper.

-
- [HL88b] C.J.M. Heemskerk and T.L.M. Luiten. Date: a benchmark for flexible assembly systems. *Assembly Automation*, 8(4):205-210, November 1988.
- [HM88] C.J.M. Heemskerk and B.R. Meijer. DIAC test BOSCH. August 1988. Intern rapport TU-Delft DIAC Project, wp1.
- [Hoc86] R. Hocken. An overview of robot programming systems. In *Annals of the CIRP*, Vol. - 35/2, pages 495-503, 1986.
- [Hol86] J. Hollingham. Flexible robot cell assembles motor transmissions. *Assembly Automation*, 6(1):15-18, February 1986.
- [HP86] V. Hayward and R.P. Paul. Robot manipulator control under UNIX RCCL: a robot control "C" library. *The International Journal of Robotics Research*, 5(4), Winter 1986.
- [HP87] D.J. Hatley and I.A. Pirbhai. *Strategies fo Real Time System Specification*. Dorset House Publishing, New York, 1987.
- [HS87] S.J. Harrington and P.J. Sackett. A study of robotic assembly systems. *Assembly Automation*, 7(3):122-126, August 1987.
- [HW86] C. Hayes and P.K. Wright. *Automated Planning in the Machining Domain*. Technical Report CMU-RI-TR, Robotics Institute, Carnegie Mellon University, Pittsburgh PA, 1986.
- [Jam88] B.J. James. *Design for Assembly, toepassing bij Signaal*. Technical Report WPS-88.010, TU-Delft, Laboratorium WPS, November 1988.
- [Jon87] P.P. Jonker, editor. *Onderzoeksplan Intelligente Montage Cel*. TU-Delft, October 1987. SPIN/FLAIR project proposal version 2-P (In Dutch).
- [Jon88] P.P. Jonker, editor. *The Architecture of the Delft Intelligent Assembly Cell*. TU-Delft, December 1988. Milestone 1 report SPIN/FLAIR-2 IV-1.1.
- [LM81] J.C. Latombe and E. Mazer. LM: a high-level programming language for controlling manipulators. In *Proceedings of the 11th International Symposium on Industrial Robots*, pages 683-690, 1981. Tokyo.

- [Loz76] T. Lozano-Pérez. *The Design of a Mechanical Assembly System*. Master's thesis, Massachusetts Institute of Technology, December 1976.
- [Loz82] T. Lozano-Pérez. *Robot Programming*. Technical Report A.I. Memo No. 698, MIT Artificial Intelligence Laboratory, December 1982.
- [Lui88] T.L.M. Luiten. *DATE: Delft Assembly Test, het ontwerp van een applikatietest voor montage-opstellingen*. Master's thesis, TU-Delft, Laboratorium WPS, February 1988. Report no. WPS-88.004.
- [LW77] L.I. Lieberman and M.A. Wesley. AUTOPASS: an automatic programming system for computer controlled mechanical assembly. *IBM Journal of Research and Development*, 21(4):321-333, July 1977.
- [MA86] C.A. Malcolm and A.P. Ambler. Some architectural implications of the use of sensors. In *Preprints of the International Conference on Intelligent Autonomous Systems*, pages 71-78, Amsterdam, December 1986.
- [Mar86] P. Martens. *Vocational use of a low cost robot for individuals with disabilities, part 1*. Master's thesis, TU-Delft, Laboratorium WPS, September 1986. Report no. WPS-86.029.
- [Mar87] P. Martens. *Vocational use of a low cost robot for individuals with disabilities, part 2*. Master's thesis, TU-Delft, Laboratorium WPS, February 1987. Report no. WPS-87.009.
- [Mar88a] J.S.P. Marauta. *Automatisch vullen van cylinders bij de firma LIPS*. Technical Report WPS-88.031, TU-Delft, Laboratorium WPS, March 1988.
- [Mar88b] J.S.P. Marauta. *Automatisering van de assemblage van cylinders bij de firma LIPS*. Master's thesis, TU-Delft, Laboratorium WPS, March 1988. Report no. WPS-88.013.
- [Mei87] N. Meijer. *Een modulaire structuur voor het off-line programmeren van Flexibele Montage Cellen*. Master's thesis, TU Delft, Laboratorium WPS, April 1987. Report no. WPS-87.021.

-
- [MGM88] MGM pneumatic brake assembly cell. Private communication with the Rofalab, TU-Delft, 1987, 1988.
- [MH88] B.R. Meijer and C.J.M. Heemskerk. DIAC test IBM. June 1988. Intern rapport TU-Delft DIAC Project, wp1.
- [MK85] A.J. Mc Killop. Robotic keybutton insertion. In *Proceedings of the 6th International Conference on Assembly Automation*, pages 267–276, IFS Ltd, 1985.
- [NN85] N.N. Assembly dominates robot applications. *The Industrial Robot*, 131–135, June 1985.
- [NN87] N.N. Assembly automation comes under the influence of robotics. *The Industrial Robot*, 45–48, March 1987.
- [NN88] N.N. Rise in installations as robots enter new areas. *The Industrial Robot*, 15(2):101–102, June 1988.
- [PAB78] R.J. Popplestone, A.P. Ambler, and I.M. Bellos. RAPT - A language for describing assemblies. *The Industrial Robot*, 5(3):131–137, 1978.
- [PAK82] R.J. Popplestone, A.P. Ambler, and K.G. Kempf. An experiment in the off-line programming of robots. In *Proceedings of the 12th International Symposium on Industrial Robots*, pages 491–503, 1982. Paris.
- [Que87] P.G. Quelle. *Toepassingsmogelijkheden van passieve vrijheidsgraden in een Flexibele Montage Cel*. Technical Report WPS-87.061, TU Delft, Laboratorium WPS, November 1987. (2 delen).
- [Ram86] H.K. Rampersad. *Path planning en collision avoidance*. Technical Report WPS-86.045, TU-Delft, Laboratorium WPS, December 1986.
- [Ram87] H.R. Rampersad. *Het testen van Montagehandelingen in Testbed-0*. Master's thesis, TU Delft, Laboratorium WPS, August 1987. Report no. WPS-87.032.

- [RC84] K. Rathmill and K. Collins. Development of an european benchmark for the comparison of assembly robot programming systems. In *Proceedings of the First Robotics Conference, Brussel*, 1984.
- [RdH86] L.N. Reijers and H.J.L.M. de Haas. *Flexibele Productie Automatisering*. Volume 3: Industriële Robots, Technische Uitgeverij De Veij Mesdagh BV, Middelburg, The Netherlands, 1986.
- [RdH88] L.N. Reijers and H.J.L.M. de Haas. *Flexibele Productie Automatisering*. Volume 1: Numerieke Besturing, Technische Uitgeverij De Veij Mesdagh BV, Middelburg, The Netherlands, 1988.
- [Rei89] L.N. Reijers. Flexibele automatisering van de montage bij serie-productie. In *Proceedings Symposium Flexibele Productie Automatisering en Informatica*, pages 50–59, April 1989.
- [Rem87] U. Rembold. Programming of industrial robots: today and in the future. In U. Rembold and K. Hörmann, editors, *Languages for Sensor-Based Control in Robotics*, Springer-Verlag Berlin Heidelberg, 1987. NATO ASI Series Vol. F29.
- [Ron88] E. Rontberg. *Een concept voor een standaard commandoset voor het beschrijven van robotbeweging*. Master's thesis, TU-Delft, Laboratorium WPS, August 1988. Report no. WPS-88.044.
- [Roo85a] B.A. Roofs. Customers to benefit from automation investment. *Assembly Automation*, 5(1):11–14, February 1985.
- [Roo85b] B.A. Roofs. Robot assembly gives flexibility in computer manufacture. *The Industrial Robot*, 169–173, September 1985.
- [Rot80] K. Roth. Systematik fester Verbindungen als Grundlage fuer ihre sinnvolle Anwendung und Weiterentwicklung. *VDI-Z*, 122(10):381–389, 1980.
- [S*87] B.E. Shimano et al. AIM: a task level control system for assembly. In *Proceedings of the 8th International Conference on Assembly Automation*, pages 187–200, IFS (Conferences) Ltd, March 1987.

-
- [SC86] R.C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56-68, Winter 1986.
- [SFD87] G. Seliger, I. Furgac, and A. Deutschländer. Set-up flexible assembly system. In *Proceedings of the 8th International Conference on Assembly Automation*, pages 289-302, IFS (Conferences) Ltd, March 1987.
- [Shi76] B. Shimano. VAL: a versatile robot programming and control system. In *Proceedings of COMPSAC 79*, pages 878-883, IEEE, 1976.
- [SHM89] T. Storm, C.J.M. Heemskerk, and P. Martens. Ontwerp en montage. *PT Industriële Automatisering*, 44(4):40-44, April 1989.
- [Sit87] Y.W. Sit. *Analyse van het grijperprobleem op de ASEA IRB-1000 opstelling in het Robotlaboratorium*. Technical Report WPS-87.055, TU Delft, Laboratorium WPS, October 1987.
- [Sny85] W.E. Snyder. *Industrial Robots: Computer Interfacing and Control*. Prentice-Hall, Inc, Englewood Cliffs, NJ 07632, 1985.
- [SSC88] R.C. Smith, M. Self, and P. Cheeseman. *Estimating Uncertain Spatial Relationships in Robotics*, pages 435-461. Elsevier Science Publishers B.V. (North-Holland), 1988.
- [TR88] R.H. Taylor and V.T. Rajan. The efficient computation of uncertainty spaces for sensor-based robot planning. In *IEEE International Workshop on Robots and Systems*, Tokyo, October 1988.
- [vBdW87] H. van Brussel and D. de Winter. Flexibele assemblagesystemen: stappen bij ontwerp, toepassing en programmering. *MB-Produktietechniek*, 53(1):9-14, January 1987.
- [vBdWT*86] H. van Brussel, D. de Winter, J. Thielemans, P. Valkenaers, and H. Claus. Introducing flexibility in assembly systems. In *Proceedings of the 16th International Symposium on Industrial Robots, Brussel*, 1986.

- [vdDH89] J.H.M. van der Drift and C.J.M. Heemskerk. Propagation of spatial uncertainties between assembly primitives. In *Proceedings of the 1989 IFAC/IFIP INCOM Conference, Madrid*, September 1989. Also: Technical report No. WPS-88.046, TU-Delft, Laboratory for Manufacturing Systems.
- [VDI75] VDI. Richtlinie 2251: *Feinwerkelemente, Verbindungen*. Beuth-Verlag, Berlin, 1975.
- [Ver85] C.G. Verweij. *Flexibele geautomatiseerde montage van remkrachtcylinders*. Master's thesis, TU-Delft, Laboratorium WPS, Februari 1985. Report no. WPS-85.001.
- [Vri88] B.I. Vries. *Stabiliteit van een stapel blokken*. Technical Report WPS-88.012, TU-Delft, Laboratorium WPS, March 1988.
- [vtE88] A.H. van 't Erve. *Generative Computer Aided Process Planning for Part Manufacturing*. PhD thesis, Universiteit Twente, January 1988.
- [vtEK86] A.H. van 't Erve and H.J.J. Kals. XPLANE, a generative computer aided process planning system for part manufacturing. In *Annals of the CIRP, Vol. - 35/2*, 1986.
- [WB88] P.K. Wright and D.A. Bourne. *Manufacturing Intelligence*. Addison-Wesley Publishing Company, Inc., 1988.

Samenvatting

Een concept voor computerondersteunde werkvoorbereiding van flexibele montage

Dit proefschrift beschrijft de ontwikkeling van een concept voor computerondersteunde werkvoorbereiding van montageprocessen op flexibele montagesystemen. Als belangrijke functies van een dergelijk werkvoorbereidingsstelsel op de langere termijn kan worden gezien:

1. Het bieden van ondersteuning bij het genereren van montage-plannen, het kiezen van apparaten om bepaalde montagehandelingen te verrichten, het bepalen van strategieën om die apparaten te gebruiken en het vaststellen van de volgorde waarin de verschillende handelingen het best plaats kunnen vinden.
2. Het ondersteunen van de keuze voor de meest bruikbare opstelling uit een set van beschikbare opstellingen en tevens het ondersteunen van het ontwerp van een (nieuwe) opstelling door het doorrekenen van het effect van variaties op die opstelling
3. Indirekte ondersteuning van het produktontwerpproces door het geven van terugkoppeling over de "maakbaarheid" van een ontwerp.

Om deze functies te vervullen moet het werkvoorbereidingssysteem de toewijzing van taken aan systeemcomponenten uitwerken, acties op verschillende abstraktieniveaus in de juiste volgorde zetten en synchroniseren, en een voorspelling doen over de verwachte kosten (voornamelijk in termen van montagetijd) op basis van simulatie, berekening en proceskennis.

Wil computerondersteunde werkvoorbereiding zinvol zijn, dan moet aan een groot aantal randvoorwaarden voldaan worden. In het werk dat beschreven is in dit proefschrift, is aangenomen dat aan die randvoorwaarden wordt voldaan. De aannames zijn:

- Het produkt is voor flexibele montage ontworpen (Design For Assembly technieken zijn toegepast)
- Het systeemontwerp is weldoordacht. Niet alle problemen hoeven op een flexibele manier opgelost te worden. Een kleine konstruktieve ingreep in de geometrie van een grijpervinger kan bijvoorbeeld een gripper geschikt maken voor het hanteren van meer dan één onderdeeltype.
- Er is voldoende flexibiliteit in het voortraject (onderdeleninkoop, fabricage) aanwezig.

Het werk heeft zich geconcentreerd op het montageproces binnen een enkele werkplek. Een hiërarchisch referentiemodel voor werkvoorbereiding in de montage wordt gepresenteerd. Het montageproces wordt hierbij opgesplitst in vier abstraktieniveaus. Een taak op batch niveau wordt via (deel)taken op produkt en onderdeel niveau op het laagste niveau opgesplitst in een verzameling primitieve handelingen, die sterk gekoppeld zijn aan fysieke apparaten.

De montageplannen worden geoptimaliseerd naar een zo kort mogelijke montagecyclustijd. Heuristische zoekalgoritmes die in verschillende stadia worden toegepast, garanderen dat goede plannen verkregen kunnen worden met een beperkte rekentijd.

Op het batch niveau worden de mogelijkheden en beperkingen van het systeem vastgelegd in de vorm van globale (strategische) beslissingen betreffende gereedschapsets en aanvoermethodes. Experimenten toonden aan dat naast de manier van gripperwisselen, ook de snelheid en layout van het transport systeem en de wijze van onderdeelaanvoer een belangrijke invloed kunnen hebben op de montage-cyclustijd.

Het toepassen van de zogenaamde sub-batch strategie kan de invloed van gripperwisseling op de montage-cyclustijd worden teruggedrongen. De sub-batch strategie houdt in dat meerdere produkten tegelijkertijd in behandeling zijn. De produkten circuleren daartoe in een produktbuffer. Elk produkt krijgt eerst een onderdeel van type 1, dan wordt één keer van gripper gewisseld. Vervolgens ontvangen alle produkten een onderdeel van type

2 etc. Op deze wijze worden de gripperwisselingen verdeeld over alle produkten in behandeling. Een zorgvuldig ontwerp van het montagesysteem moet voorkomen dat pallet-wisseltijden dit gunstige effect niet ongedaan maken.

Op het produkt niveau wordt een ruwe verzameling van mogelijke montagevolgordes direkt uit de produktbeschrijving afgeleid (een generatief proces). Om de complexiteit van de volgordebepaling te reduceren is een nieuwe techniek voor het groeperen van onderdelen ontwikkeld (clusteren). De groepen of clusters komen overeen met bepaalde structuren in het produkt, zoals een laag en een stapeling (sandwich structuur). Alle onderdelen in een cluster hebben dezelfde (hoofd)montagerichting, en worden verondersteld direkt na elkaar gemonteerd te worden.

In samenhang met de clusterdefinitie is een nieuwe, gelaagde methode voor het representeren van montagevolgordes ontworpen (het Layered Assembly State Transition Diagram). Het LASTD is kompakt en flexibel van opbouw, en kan zowel bij het bepalen van mogelijke montagevolgordes als tijdens het aansturen van het montageproces gebruikt worden. Het LASTD geeft zowel de samenstelling van elke subassembly als elke relevante volgorde van handelingen expliciet weer.

Heuristische regels ("rules") worden gebruikt om het uit het ruwe volgorde plan instabiele tussentoestanden te elimineren. Een zoektechniek gebaseerd op het A* algoritme werkt tenslotte het plan tot in de details uit, waarbij een beroep wordt gedaan op lager niveau planners.

Door middel van experimenten werd aangetoond dat, mits een goede, gedetailleerde beschrijving van het product beschikbaar is, het mogelijk is om volgorde-plannen automatisch af te leiden, met een beperkte rekentijd (CPU-tijd in de orde van seconden).

Het onderdeel niveau vormt een interface tussen de abstracte "monteer onderdeel x" akties van het produkt niveau en de sterk apparaat-gebonden akties op primitieve niveau. Het zoekproces naar een optimale volgorde van akties op het onderdeelniveau wordt geleid door een door een systeem-programmeur gedefinieerd "part script", een soort draaiboek waarin alle primitieve handelingen met elkaar in verband worden gebracht. Tijdens het zoekproces wordt steeds de meest belovende handeling geëxpandeerd, en wordt nagegaan welke apparaten het meest geschikt zijn.

Tijdens het uitvoeren van de werkvoorbereiding wordt op onderdeel niveau in een systeemmodel een aktueel beeld van alle positie-onzekerheden bij-

gehouden. Als na een aantal handelingen de totale onzekerheid te hoog is opgelopen, wordt automatisch een positiemeting ingelast.

Het primitieve niveau bestaat uit een soort bibliotheek van apparaat-gebonden akties. Er wordt een onderscheid gemaakt tussen selektie- en aktie-primitieven. Selektie primitieven ondersteunen de werkvoorbereiding door telkens voor een enkele aktie het meest geschikte apparaat uit te zoeken. Wanneer de planner in een later stadium constateert dat het geselecteerde apparaat toch niet werkt (bijvoorbeeld omdat het te onnauwkeurig is), dan komt men automatisch terug bij de selektie aktie en wordt een "tweede keus" gemaakt.

De voorgestelde aanpak van de werkvoorbereiding is erg flexibel, omdat op elk abstraktieniveau alternatieve strategieën worden gegenereerd en geëvalueerd. Deze aanpak is algemeen bruikbaar, zoals gedemonstreerd werd met een prototype werkvoorbereidingssysteem, dat plannen genereerde voor verschillende produkten op verschillende montagesystemen. De gegenereerde plannen bleken executeerbaar.

Bij het ontwikkelen van de software is gebruik gemaakt van een gestructureerde modelleertechniek (gebaseerd op de methode van Yourdon), ondersteund door een pakket op de IBM-PC (PROMOD). Het gebruik van deze ontwikkelmethode heeft geleid tot een goed gestructureerd en relatief eenvoudig overdraagbaar software ontwerp.

De produkt en systeem modellen zijn opgeslagen in een relationeel database pakket (INGRES) gebouwd. Nieuwe eisen met betrekking tot de gewenste opbouw van het produktmodel zijn opgesteld. Belangrijke nieuwe datastructuren zijn het onderdeel-positie- en het onderdeel-relatie-netwerk.

Op verschillende "echte" montagesystemen zijn experimenten uitgevoerd. Diverse nieuwe apparaten en konstrukties zijn daarvoor ontwikkeld en gebouwd. Deze geïntegreerde aanpak heeft het hele project een extra waarde gegeven. In sommige gevallen is het zwaartepunt van bepaalde experimenten verschoven, en het belang van verschillende ontwerpbeslissingen werd nog eens benadrukt. Het verhoogde tevens de motivatie van bij het project betrokken personen en het verhoogde ook de "zichtbaarheid" van het project voor de buitenwereld. Aan de andere kant vergde de ontwikkeling van de testbedden een hoge investering in termen van manuren en investeringen. De "engineering" fase bleek gecompliceerder en tijdrovender dan verwacht.

Het in dit proefschrift beschreven werk had een verkennend karakter en had mede tot doel het aangeven van zwaartepunten voor verder onderzoek. Het

onderzoek is ook in dit opzicht succesvol geweest. Het heeft onder andere bijgedragen aan de definitie en de opzet het "Delft Intelligent Assembly Cell" projekt (DIAC). De ontwikkelde hard- en software prototypes zullen worden gebruikt en verder ontwikkeld in DIAC.

Glossary

Assembly Used either to refer to the process of bringing together parts (of a product) to form a unit, or to the result thereof (the parts so assembled). In a restricted sense, assembly only covers the bringing together and joining of parts (see introduction), in a broader sense the term also covers some pre- and post-post-processing processes like cleaning and inspection.

ASTD Assembly State Transition Diagram.

AT Approximate Transform, describes the relation between two frames in a geometric model of the assembly system. One part describes the nominal position and orientation another part describes the uncertainty in position and orientation.

Batch A set of identical products, assembled in one (large) sequence of assembly operations executed on one or more assembly workstations in a single assembly system. No major adjustments to the assembly system take place during production of a batch.

BOM Bill Of Material, a list that is attached to the composition drawing of a product, describing all parts contained in a product. Each entry in the list describes a different part type. The BOM can hold information on the quantity and material of the parts, the part size and a reference to standards.

Cluster A group of parts that belong together logically because they have some assembly features in common, because of which they normally will be assembled as a group. The combined parts do not necessarily form a stable subassembly. A key assumption is that once the assembly of a group is started, the sequence may not be interrupted by parts from outside that group.

CTA Current To Assembled (transition in ASTD or PSTD).

CTL Current To Leaf (transition in ASTD or PSTD).

DIAC Delft Intelligent Assembly Cell, a cooperative research project involving eight research groups from four faculties of the Delft University of Technology. It was officially started in december 1987, lasting four years. With a total budget of about 10 Mf, DIAC is sponsored by the Dutch SPIN/FLAIR program and by various industrial partners /citediacrapport.

DOF Degree(s) Of Freedom. A solid part in a free (three dimensional or 3D) space has six degrees of freedom (three translations and three rotations). A DOF can also be relative, e.g. two parts touching by a common (flat) surface have 3 relative DOF. Sometimes a distinction will be made between positive and negative directions of motion, artificially doubling the DOF of a part in 3D space to twelve.

Feature A distinct or outstanding quality of an object, used to refer to a surface or a set of surfaces involved in establishing a connection with another object, or to localize the object in space.

Frame Short for "frame of reference". Characteristic whereby objects are localized in space.

LTA Leaf To Assembled (transition in ASTD or PSTD).

Part An essential, separate element from which a product is constructed.

PLC Programmable Logic Controller

Primitive Basic operation in the assembly process, involving only one actuator or a limited (and fixed) set of actuators, in a pre-described way. In this work, a distinction is made between planning (selection) and action (simulation or execution) primitives.

Product The result of a complete sequence of assembly operations, combining a (large) set of parts into one functional unit, as specified by the product model.

PROMOD A tool for Software Engineering, helpful for creating high level documentation in the form of data flow diagrams, a data dictionary etc. In the project we used PROMOD V1.06, running on an IBM/PC-AT compatible computer under the MS-DOS operating system.

PSTD Part State Transition Diagram.

PTC Parent To Current (transition in ASTD or PSTD).

Relation (Sometimes Part_Relation:) The result of one or more assembly operations, reducing the relative degrees of freedom of one part of a construction with respect to other parts. Relation is also used

SPIN/FLAIR SPIN stands for "StimuleringsPlan Informatica in Nederland", a national program for stimulation of research in the area of computer science, sponsored by the Ministry of Economic Affairs. FLAIR stands for "Flexibele Automatisering met Industriële Robots", and sponsors research in Industrial Robotics. At first a separate funding program, FLAIR was incorporated into the SPIN program in 1986.

STW Stichting voor de Technische Wetenschappen

TCP Tool Center Point

TES Tool Exchange System

List of Figures

1.1	Functional Model of a Production Company	4
1.2	Five types of flexibility playing a role in the life cycle of a production system	6
1.3	Application of Industrial Robots in Germany (Source: IPA) .	8
1.4	Yearly production volume	11
1.5	Total number of parts per product	12
1.6	Number of part types per product	13
1.7	Number of part types per arm	14
1.8	The assembly process as a black box	15
2.1	A product oriented decomposition of the assembly process into four levels of abstraction	29
2.2	The assembly process as a sequence of join operations.	30
2.3	Elementary sequence of actions at the part level	32
2.4	Context of the Assembly Process Planner	36
2.5	Decomposition the Assembly Process Planner	38
3.1	Determination of part position between manufacturing and assembly	41
3.2	Part and product flow according traditional concepts	47
3.3	Pallet flow in a workstation that supports a sub-batch strategy	48
3.4	Layout of the IVF Mark II system	49

3.5	Layout of the ASCO manual assembly workstation	51
4.1	Example product with 5 parts	54
4.2	The influence of accessibility and stability	57
4.3	Triangle table for the 5 part product.	60
4.4	Three different precedence diagrams for the 5 part product. .	61
4.5	AND/OR graph for the 5 part product.	63
5.1	Global structure of the Sequence Planner.	91
5.2	The Cranfield Benchmark product.	94
5.3	Relation network for the simplified Cranfield Benchmark (CB9). 95	
5.4	Example of a product with a stack cluster	98
5.5	Example of a product with a layer cluster	99
5.6	Example of a product with a sort cluster	100
5.7	Example of a product with a combi cluster	100
5.8	Cluster recognition cycle	102
5.9	Sequence expansion	109
5.10	Relation network for the 9 part version of the Cranfield Benchmark (CB9)	110
5.11	Relation network for CB9 after reduction of SP_1	111
5.12	Relation network for CB9 after recognition of two sort clusters	112
5.13	Top level LASTD for CB9 found with cluster recognition and the disassembly approach	113
5.14	Top level LASTD for CB9 after first two stability rules	114
5.15	Total LASTD for CB9	116
6.1	Part-script and part level plan generation	119
6.2	Global structure of the Part Level Planner	121
6.3	Modelling the assembly environment with a network of frames	123
6.4	A typical action that can be specified in RAPT	124

6.5	An example of accumulating uncertainty in the assembly environment	128
6.6	Reducing uncertainty in the assembly environment	128
6.7	Definition of a relation	132
6.8	Conservative approximation of dependent position parameters	134
6.9	The assembly environment modeled as a network of $AT's$. .	135
6.10	Comparing the uncertainty of a critical relation with a norm	136
6.11	Propagation of position uncertainties 1	138
6.12	Propagation of position uncertainties 2	139
6.13	Compound of two relations	141
6.14	Merge of two relations	141
6.15	Eliminating a dead-end	142
6.16	Reduction of a loop structure	143
6.17	Inverting a relation	143
6.18	Two dimensional example of a deterministic compound . . .	145
6.19	Example of a deterministic merge in two dimensions	146
7.1	Global structure of the grip planner	155
7.2	Measurement of side plate position with three line scans . .	159
8.1	Parts of the Cranfield Benchmark on the pallet	167
8.2	The Cranfield Benchmark product in assembled form	168
8.3	DATE: The Delft Assembly TEST product	171
8.4	Layout of the ASEA test site	177
8.5	Control Architecture of the ASEA site	179
8.6	Layout of the ROBOT VISION (ROVIS) test site	180
8.7	Control structure of the ROVIS test site	181
8.8	Pallet Flow in the ASEA site using the sub-batch principle .	185

8.9	Total LASTD for the nine part version of the Cranfield Benchmark.	189
8.10	The four "cheapest" sequences for assembling the nine part version of the Cranfield Benchmark with the TES based set of grippers.	192
8.11	DIAC specification product, mainly consisting of rectangular parts (Medusa 3D model)	193
8.12	Uncertain relations in a simple assembly environment	195
B.1	Context diagram	247
B.2	Top level: Make_Plan	248
B.3	The product level sequence planner: Plan_Product	248
B.4	Cluster generation: Make_Cluster	249
B.5	EXpansion of Sequences: Expand_and_Verify	249
B.6	The part level planner: Plan_Part	250
B.7	Propagation of position uncertainties: Check_Position_Propagation	250
B.8	The primitive level: Select_Primitive	251
B.9	The primitive level: Select_Grip	251
C.1	Pallet Flow in the ASEA site using the sub-batch principle .	259

List of Tables

2.1	The evolution of CAPP Methods	20
3.1	Comparison of parts feeding methods	43
3.2	Comparison of tool change strategies	46
5.1	Part list for the Cranfield Benchmark	94
5.2	The effect of the number of clusters on the complexity of sequence planning	104
5.3	The effect of cluster size on the complexity of sequence planning	105
5.4	The influence of stack cluster size on the complexity of the LASTD	106
6.1	Examples of sensor information processing strategies for var- ious types of sensors	131
7.1	Griprimitives supported in the planner prototype	154
8.1	Programming time for the four peg test	169
8.2	Structure of this chapter	174
8.3	Robot interface in distributed architecture	182
8.4	Overview of batch level experiments	183
8.5	Timing results of the batch level experiments	183
8.6	Pallet transport times	187

8.7	Sequence plan computation times for various products	188
8.8	Overview of gripper suitability for Cranfield Benchmark parts.	191
A.1	Main characteristics of assembly systems described in literature	242
A.2	Main characteristics of assembly systems described in literature	243
C.1	Overview of the batch level experiments	253
C.2	Description of the two gripper sets	254
C.3	Timing of experiment 1	255
C.4	Timing of experiment 2	255
C.5	Timing of experiment 3	256
C.6	Timing of experiment 4	257
C.7	Description of pallets used for the sub-batch experiments . .	258
C.8	Description of the stations in the VTS used for the sub-batch experiments	258
C.9	Description of the stations in the VTS used for the sub-batch experiments	260
C.10	Timing of experiment 5	262
C.11	Computed action times of experiment 5	263
C.12	Timing of experiment 6	264
C.13	Computed action times of experiment 6	265

Appendix A

Flexible Assembly Systems in Literature

The tables A.1 and A.2 on the following pages give an overview of the main characteristics of assembly systems as described in literature. Thirty-nine of the forty-seven systems listed here were described as part of a study reported on in Assembly Automation [HS87]. For seven systems of the thirty-nine, a second reference was found, each time with a well-corresponding description. However, not all systems from the study were included. Excluded were all systems that did not conform to one of the two basic architectures: cell or line. Systems whose structure was not described and systems with a "cell/line" structure were excluded. Also "cell"-structured systems with more than two robots and "line"-structured systems with only one robot were excluded.

Other references that were consulted are indicated in the tables.

No.	Product	C/L	R	CT	TP	TDP	Reference
1.	Cylinder head	L	8		25	3	[HS87]
2.	Turbo charger	L	8		8	7	[HS87]
3.	Contactor switch	C	1	4.0	21	9	[HS87]
4.	Heater	L	3	0.15	8	8	[HS87]
5.	Terminal block	C	2	0.6	27	3	[HS87]
6.	Disk drive	C	1		8	7	[HS87]
7.	Word processor	L	3		50	19	[HS87]
8.	Diskettes	C	1	12.0	41	4	[HS87]
9.	Pressure transmitter	C	1		13	6	[HS87]
10.	Keyboard caps	C	1	1.0	55	6	[HS87]
11.	Fan motor	L	13		13	10	[HS87]
12.	Ball valve	C	1	1.2	6	5	[HS87]
13.	Cylinder head	L	3		97	6	[HS87]
14.	VTR mechanism	L	11		60	30	[HS87]
15.	Contactor	C	2	0.4	12	10	[HS87]
16.	Moped crankcase	L	18		56	10	[HS87]
17.	Windscreen	L	4	0.17	3	3	[HS87]
18.	Engine subassembly	L	5		38	12	[HS87]
19.	Dot matrix printer	L	48	0.3	44	26	[HS87]
							[Har85]
20.	Cylinder head	L	8	0.92	147	17	[HS87]
21.	Mascara bottle	C	2	0.1	3	3	[HS87]
22.	Lipstick	C	2	0.06	4	4	[HS87]
23.	Rear windscreen	L	4	1.5	2	2	[HS87]
24.	Computer subassembly	C	2		10	6	[HS87]
25.	Cylinder head	L	3	12.0	73	4	[HS87]
26.	Watch casing	L	11	0.05	12	11	[HS87]
							[NN85]
C/L System structure, C: Cell, L: Line R Number of robots in the system CT Cycle Time (Minutes per product) TP Total number of parts in the product TDP Total number of different parts in the product							

Table A.1: Main characteristics of assembly systems described in literature

No.	Product	C/L	R	CT	TP	TDP	Reference
27.	PCB television	L	11		100	20	[HS87]
28.	Windscreen	L	6		3	3	[HS87]
29.	Switch block	C	1		19	6	[HS87]
30.	Car radio	L	3		11	10	[HS87]
31.	PCB mainframe comp.	C	1	0.5	5	5	[HS87] [Roo85b]
32.	Cylinder head	L	16	2.37	91	12	[HS87]
33.	Car flywheel	C	1	0.33	10	2	[HS87]
34.	Car console	C	2		10	3	[HS87]
35.	Robot linkage	C	2	8.0	39	11	[HS87] [Har86b]
36.	Pump	C	1		4	3	[HS87]
37.	Connector rod	C	2	1.22	4	3	[HS87]
38.	Diesel inject. pump	L	7	0.53	8	7	[HS87] [Har86a]
39.	Disk file	C	1		10	6	[HS87]
40.	Gas valve	L	4	1.04	20	20	[Bou87a]
41.	Cylinder head	L	8	1.1	70	10	[Bad85] [Roo85a]
42.	Gear box	L	4	0.25	10	10	[Hol86]
43.	Keyboard	L	5	1.5	100	10	[MK85]
44.	Type writer part	C	1	0.5	6	6	[BvdW87]
45.	Cylinder head	L	4	1.2	56	11	[NN87]
46.	Shaving head	L	3	0.06	5	3	[CIM88]
47.	Air brake cylinder	C	1	0.9	5	5	[MGM88]
C/L System structure, C: Cell, L: Line R Number of robots in the system CT Cycle Time (Minutes per product) TP Total number of parts in the product TDP Total number of different parts in the product							

Table A.2: Main characteristics of assembly systems described in literature

Appendix B

Dataflow Diagrams and Data Tree

B.1 How to read a ProMod dataflow diagram

Dataflow diagrams contain the following elements:

Nodes A Node (or Process) is depicted in the dataflow diagram by a circle. It transforms information from incoming dataflows to outgoing dataflows. If the process is a high level abstraction, one can "zoom in" to find a child diagram

Dataflows A Dataflow is depicted in the diagram by an arrow. Dataflows represent the information flowing to and from processes.

Terminators A terminator represents a connection to another system outside the current scope, e.g. a user interface. Terminators are most common at the highest abstraction level (the context diagram).

Stores A store is depicted by two parallel horizontal lines. It represents the (long term) storage of non-volatile information. Typical implementation occurs in the form of a file or a database.

Dataflows can contain Splits and Merges: a dataflow can split up into subflows, or several subflows can combine into a single "superflow". Within a single dataflow diagram, ProMod only allows complete splits and merges,

i.e. all subflows contain the same information as the main flow. Real subdivision of dataflows may only take place between parent and child diagrams. The reader thus has to refer to the data dictionary to see how the dataflows split up.

B.2 Dataflow Diagrams

B.3 Data Tree

This section is added only to provide a first impression. The complete description of processes, dataflows etc. is provided in a separate document.

```
ASTD_PTC_C_and_T
=====
.  ASTD_Parent_State_ID
.  .  ASTD_State_ID
.  .  .  Integer
+  ASTD_Current_State_ID
.  .  ASTD_State_ID  -->
+  ASTD_PTC_Cost
.  .  ASTD_Trans_Cost
.  .  .  Float
```

Make_Cluster

LEVEL: 1.2.2	LAST UPDATE: 19-DEC-1989 18: 00
PROJECT: makeplan	LAST ANALYSIS: 16-MAR-1990 13: 51

Expend_and_Verify

LEVEL: 1.2.5	LAST UPDATE: 15-MAR-1990 14: 42
PROJECT: makeplan	LAST ANALYSIS: 16-MAR-1990 13: 51

ASTD_Successors

=====

```

.   ASTD_CTL_Trans
.   .   ASTD_Current_State_ID
.   .   .   ASTD_State_ID
.   .   .   .   Integer
+   .   ASTD_Leaf_State_ID
.   .   .   ASTD_State_ID  -->
+   .   ASTD_Leaf_State
.   .   .   ASTD_Leaf_State_ID  -->
+   .   ASTD_State_Description
.   .   .   {ASTD_Assembled_Part_Group}
.   .   .   .   {ASTD_Assembled_Part}
.   .   .   .   .   Part_Id
.   .   .   .   .   .   Integer  -->
+   .   {ASTD_Loose_Part}
.   .   .   Part_Id  -->

```

Batch_Offer

=====

```

.   Batch_ID
+   .   Batch_Task_Graph
.   .   {Batch_Strategy_and_Cost}
.   .   .   Batch_Strategy
.   .   .   .   Feed_Strategy
.   .   .   .   +   Grasp_Strategy
.   .   .   .   +   Move_Strategy
.   .   .   .   +   Mount_Strategy
.   .   .   .   +   Sense_Strategy
.   .   .   .   +   Tool_Strategy
+   .   .   Batch_Cost_Estimate

```

Batch_Request

=====

```

.   Product_ID
.   .   Integer
+   .   Product_Quantity
+   .   System_Layout

```


=====

see STALASTD

Command

=====

. Expanded_Primitive

Critical_Relation

=====

. Object_From_ID

+ Object_To_ID

Detected_Clusters

=====

. Sort_Clusters

. {Cluster}

. Cluster_ID

. Integer

+ {Part_Id}

. Integer -->

+ {Child_Cluster}

. {Cluster} -->

+ Stack_Clusters

. {Cluster} -->

+ Mix_Layer_Clusters

. {Cluster} -->

+ Sort_Layer_Clusters

. {Cluster} -->

+ Combi_Clusters

. {Cluster} -->

EASTD

=====

. ASTD

. ASTD_ID

. Integer

+ {ASTD_Element}

. {ASTD_State}

. ASTD_State_ID

```

    . Integer -->
+ {ASTD_Prev_State_Pointer}
    . ASTD_State_Pointer
+ {ASTD_Next_State_Pointer}
    . ASTD_State_Pointer -->
+ ASTD_PTC_Cost
    . ASTD_Trans_Cost
      . Float
+ ASTD_STC_Cost
    . ASTD_Trans_Cost -->
+ ASTD_CTA_Cost
    . ASTD_Trans_Cost -->
+ ASTD_STA_Cost
    . ASTD_Trans_Cost -->
+ ASTD_State_Description
    . {ASTD_Assembled_Part_Group}
      . {ASTD_Assembled_Part}
        . Part_Id
          . Integer -->
      + {ASTD_Loose_Part}
        . Part_Id -->
+ {ASTD_Trans}
    . ASTD_Trans_ID
    + ASTD_From_State_Pointer
      . ASTD_State_Pointer -->
    + ASTD_To_State_Pointer
      . ASTD_State_Pointer -->
    + {ASTD_Prev_Trans_Pointer}
      . ASTD_Trans_Pointer
    + {ASTD_Next_Trans_Pointer}
      . ASTD_Trans_Pointer -->
    + ASTD_Trans_Description

```

```
free_griprimitives
```

```
=====
```

```

    . {free_grprs}
      . {free_grpr}
        . [free_poly_grpr
          . griprimitive
        |free_spp_grpr]
          . griprimitive -->

```

Part_Offer

=====

- . Part_Id
 - . Integer
- + PSTD
 - . Part_Id -->
 - + {PSTD_State}
 - . {Approx_Transform}
 - . Nominal_Position
 - . Frame
 - + Uncertainty_Vector
 - . {Float}
 - + {PSTD_Trans}
 - . PSTD_Trans_ID
 - + PSTD_Trans_Type
 - + PSTD_PTC_Cost
 - + PSTD_STC_Cost
 - . PSTD_Trans_ID -->
 - + PSTD_STC_Trans_Cost
 - + PSTD_CTA_Cost
 - . PSTD_Trans_ID -->
 - + PSTD_CTA_Trans_Cost

Part_Request

=====

- . Part_Id
 - . Integer
- + ASTD_State
- see EASTD

Primitive_Cost

=====

- . Time
- + Chance_Success

Product_Offer

=====

- . Product_ID
 - . Integer
- + ASTD

see EASTD

Product_Request

=====

- . Product_ID
 - . Integer
- + Batch_Strategy
 - . Feed_Strategy
 - + Grasp_Strategy
 - + Move_Strategy
 - + Mount_Strategy
 - + Sense_Strategy
 - + Tool_Strategy

Reduced_Part_Rel_Network

=====

- . {Part_Rel_Network_Element}
 - . Part_Rel_ID
 - + Part_From_ID
 - . Part_Id
 - . Integer
 - + Part_To_ID
 - . Part_Id -->
 - + Part_Rel_Type
 - + Part_Rel_Pos
 - . Part_Rel_Pos_Reference
 - . Frame_ID
 - + Part_Rel_Pos_Frame
 - . Frame

Selected_ASTD_PTC_Trans

=====

- . ASTD_Parent_State_ID
 - . ASTD_State_ID
 - . Integer
- + ASTD_Current_State_ID
 - . ASTD_State_ID -->

Selected_ASTD_State

```

=====
.   ASTD_State_ID
.   Integer

```

Select_Offer

```

=====
.   [Feed_Offer
.   .   Feeder
.   +   Feed_Strategy
|Grip_Offer
.   .   Grip
.   .   supported_grip
.   .   + {gripper_value}
.   .   .   Float
.   .   + strategy
.   .   + {grip_value}
+   Bad_Part
.   .   [no_grip
.   .   .   Integer
.   .   |no_gripprimitive
.   .   .   Integer -->
.   .   |no_gripper]
.   .   .   Integer -->
|Move_Offer
.   .   [Moves
.   .   .   {Move}
.   .   .   .   Robot_ID
.   .   .   + Path
.   .   .   + Move_Cost
.   .   |Bad_Move]
|Sense_Offer]
.   .   Sensor_ID
.   .   .   Integer -->
+   Sensor_Work_Area
+   Sensor_Cost
.   .   Float -->
+   Sensor_Position
.   .   Approx_Transform
.   .   .   Nominal_Position
.   .   .   .   Frame
.   .   + Uncertainty_Vector
.   .   .   {Float} -->

```

Select_Request

=====

```

. [Feed_Request
  . Part_Id
    . Integer
|Grip_Request
  . Part_Id -->
  + Avail_Grippers
    . {avail_gripper}
      . gripper
        . store
          + gripper_exchange_time
          + gripper_type
          + {scalar}
          + sensors
          + gripper_dimension
|Move_Request
  . ASTD_State_Description
    . {ASTD_Assembled_Part_Group}
      . {ASTD_Assembled_Part}
        . Part_Id -->
      + {ASTD_Loose_Part}
        . Part_Id -->
  + Part_Id -->
  + Grips
  + Feeds
  + Mount_Pos
|Sense_Request]
  . Part_Id -->
  + Part_Pos

```

Sensor_Status_Update

=====

```

. Cell_Status
  . Approx_Transforms
    . {Approx_Transform}
      . Nominal_Position
        . Frame
      + Uncertainty_Vector
        . {Float}

```

=====

```

.   LASTD
.   {LASTD_Element}
.   LASTD_ID
.   Integer
+ {LASTD_State}
.   LASTD_State_ID
.   Integer -->
+ {LASTD_Prev_State_Pointer}
+ {LASTD_Next_State_Pointer}
+ {LASTD_Trans}
.   LASTD_Trans_ID
.   Integer -->
+ LASTD_From_State_Pointer
+ LASTD_To_State_Pointer
+ {LASTD_Prev_Trans_Pointer}
+ {LASTD_Next_Trans_Pointer}
+ LASTD_Trans_Description
.   LASTD_Child_Or_Part
.   [Child
    |Part]
+ [LASTD_Child_ID
.   LASTD_ID -->
|Part_Id]
.   Integer -->

```

sup_grips

=====

```

.   {sup_grip}
.   sup_poly_grips
.   {griprimitive}
+ {gripper}
.   store
+ gripper_exchange_time
+ gripper_type
+ {scalar}
+ sensors
+ gripper_dimension
+ sup_spp_grips
.   {griprimitive} -->
+ {gripper} -->

```

World_Model

=====

```

.   Product_Model
.   {Product_Model_Element}
.   Product_ID
.   Integer
+   BOM
.   {BOM_Element}
.   Part_Id
.   Integer -->
+   Part_Name
.   String
+   Part_Type
.   String -->
+   Part_Unique
.   BOOLEAN
+   Part_Material
.   String -->
+   Part_Rel_Network
.   {Part_Rel_Network_Element}
.   Part_Rel_ID
+   Part_From_ID
.   Part_Id -->
+   Part_To_ID
.   Part_Id -->
+   Part_Rel_Type
+   Part_Rel_Pos
.   Part_Rel_Pos_Reference
.   Frame_ID
+   Part_Rel_Pos_Frame
.   Frame
+   Part_Geometry
.   CAD_Model
.   Not_Defined
+   Part_Pos
+   UDF_Subassies
.   {Cluster}
.   Cluster_ID
.   Integer -->
+   {Part_Id} -->
+   {Child_Cluster}
.   {Cluster} -->

```



```
+ Process_Model
.   Standard_PSTD_Cost_Estimates
.     {Standard_PSTD_Cost_Estimate}
+ Part_Script
.   Not_Defined  -->
+ Non_Free_Feeder_Surfaces
.   {surface}
+ Non_Free_Mount_Surfaces
.   {surface}  -->
+ criteria
.   grpr_criteria
.     grpr_evaluation_procedures
.       {procedures}
.     + grpr_criteria_weight
.       {Integer}  -->
.     + grpr_limits
.       {Float}
.   + gripper_criteria
.     gripper_evaluation_procedures
.       {procedures}  -->
.     + gripper_criteria_weight
.       {Float}  -->
.     + gripper_limits
.       {Float}  -->
+ System_Model
.   {Sys_Elem}
.     Sys_Elem_ID
.       Integer  -->
.     + Sys_Elem_Name
.       String  -->
.     + Sys_Elem_Type
.       String  -->
.     + Sys_Elem_Geometry
.       CAD_Model  -->
.     + Sys_Elem_Pos
.       Space_Pos
.         Pos_Reference
.           Frame_ID  -->
.         + Frame  -->
```


Appendix C

Timing tables of batch level experiments

Experiments at the batch level were executed to examine the influence of gripper exchange strategies, parts feeding strategies and the sub-batch strategy on the assembly cycle time. Table C.1 provides an overview of the main experiments.

Exp.	Product	Tools	Feeding	Transport	Speed
1.	CB9	MG	Kit	-	20%
2.	CB9	MG	Kit	-	100%
3.	CB9	TES	Kit	-	20%
4.	CB9	TES	Kit	-	100%
5.	6*CB9	TES	Pallet	Sub-batch	20%
6.	6*CB9	TES	Pallet	Sub-batch	100%
Site	Asea				
Date	December 1989 / January 1990				
CB9	Nine part version of the Cranfield Benchmark				
MG	Multigripper				
TES	Tool exchange system				

Table C.1: Overview of the batch level experiments

All experiments were done with the nine part version of the Cranfield Bench-

mark in the Asea site. For kit assembly the benchmark base plate was mounted on a pallet that was stationed in station 3. For the experiments with the sub-batch strategy, a set of six benchmark products was created. The use of multigripper based grippers was compared with the use of TES (Tool exchange System) based grippers. A description of the two gripper sets is given in table C.2. All experiments were executed at 20% and 100% of the robot speed.

Part	Multigripper number	TES gripper number
Side plates	1	1
Pegs	2	2
Locking pins	-	2
Shaft	2	3
Handle	3	3
Block	3	3

Table C.2: *Description of the two gripper sets*

C.1 Multigripper kit assembly

The first two experiments concerned the assembly of a single benchmark product (CB9) with a multigripper. Timing results are presented in tables C.3 and tables C.4. The parts are fed in a kit (i.e. on the benchmark base plate) and the robot executed the program with speed settings of 20% and 100% respectively. At 20%, the complete product was assembled in 77 seconds, implying an average assembly time of 8.4 seconds per part. For reasons of comparison, also the average assembly time of a single peg is computed. Under the conditions mentioned, an average of 8 seconds was measured.

C.2 Kit assembly with TES mounted grippers

Experiments 3 and 4 concern the assembly of a single (kit-fed) benchmark product (CB9) with TES mounted grippers. A description of the grippers in the TES is given in table C.2. The results are described in tables C.5

Action	Time	Remark
Start	16.22.26	Robot initialized and above kit
End mounting Side Plate 1	22.36	
End mounting Peg 1	22.46	
End mounting Peg 2	22.54	
End mounting Peg 3	23.01	
End mounting Peg 4	23.08	
End mounting Shaft	23.16	
End mounting Handle	23.25	
End mounting Block (Sppce)	23.34	
End mounting Side Plate 2	23.43	
Total time	77 s	For 9 parts

Table C.3: *Timing of experiment 1: Kit assembly with a multigripper and the robot speed at 20%*

Action	Time	Remark
Start	16.24.46	
End mounting Side Plate 1	24.53	
End mounting Peg 1	24.58	
End mounting Peg 2	25.03	
End mounting Peg 3	25.07	
End mounting Peg 4	25.11	
End mounting Shaft	25.16	
End mounting Handle	25.23	
End mounting Block (Sppce)	25.30	Block jammed on grasp
End mounting Side Plate 2	25.34	
Total	48 s	For 9 parts

Table C.4: *Timing of experiment 2: Kit assembly with a multigripper and the robot speed at 100%*

and C.6. The timing results show that the total assembly time is much larger than with the multigripper (181 seconds vs. 77 seconds at 20%). The cause of the delay is the long gripper exchange time.

Action	Time	Remark
Start	09.48.00	Robot above rack no gripper
End pickup gripper 1	48.11	Includes 5 s rack to kit
End mounting Side Plate 1	48.24	
End pickup gripper 2	48.54	
End mounting Peg 1	49.05	
End mounting Peg 2	49.12	
End mounting Peg 3	49.19	
End mounting Peg 4	49.25	
End pickup gripper 3	50.00	
End mounting Shaft	50.11	
End mounting Handle	50.19	
End mounting Block (Sppce)	50.26	
End pickup gripper 1	51.00	
End mounting Side Plate 2	51.12	
End restore gripper 1	51.29	
Total assembly time	181 s	Excl. first gripper change
Tool exchange time	99 s	
Rack to kit motion time	20 s	Estimated overhead
Net assembly time	62 s	

Table C.5: *Timing of experiment 3: Kit assembly with TES based grippers and the robot speed at 20%*

C.3 Sub-batch assembly with TES mounted grippers

Experiments 5 and 6 concern the assembly of six benchmark products (6 * CB9) with TES mounted grippers. A description of the grippers in the TES is given in table C.2. Table C.7 describes the way the pallets are filled. Figure C.1 gives an overview of the physical layout of the Asea site, table C.8

Action	Time	Remark
Start	10.04.30	Robot above rack with gripper
Start pickup Side Plate 1	04.34	
End mounting Side Plate 1	04.42	
End pickup gripper 2	04.58	
End mounting Peg 1	05.07	Incl. 4 s for rack to kit motion
End mounting Peg 2	05.10	
End mounting Peg 3	05.14	
End mounting Peg 4	05.17	
End pickup gripper 3	05.35	
End mounting Shaft	05.43	
End mounting Handle	05.48	
End mounting Block (Sppce)	05.52	
End pickup gripper 1	06.10	
End mounting Side Plate 2	06.18	
End restore gripper 1	06.28	
Total assembly time	104 s	Excl. first rack to kit motion
Tool exchange time	52 s	
Rack to kit motion	12 s	Estimated overhead
Net assembly time	40 s	

Table C.6: *Timing of experiment 4: Kit assembly with TES based grippers and the robot speed at 100%*

provides a description of the stations, and table C.9 describes the position of all pallets in the VTS at the beginning of the sub-batch tests. The timing results are described in tables C.10 and C.12.

Pallet number	Contains
1.	Assembly fixtures for two benchmark products
2.	Assembly fixtures for two benchmark products
3.	Assembly fixtures for two benchmark products
5.	Six Side Plates (pallet is refilled manually and also used for top side plates)
8.	6 * 4 = 24 Spacer Pegs (room for more, can also host locking pins)
14.	6* Shaft 6* Handle (Lever) 6* Spacer Block (Spacing piece)

Table C.7: *Description of pallets used for the sub-batch experiments*

Station number	Name
1.	Bosch station (in-line buffer)
1a.	Buffer for Bosch station
2.	Manual station
2a.	Buffer for manual station
3.	Asea product position
3a.	Buffer for station number 3.
4.	Pallet buffer area
5.	Asea part position

Table C.8: *Description of the stations in the VTS used for the sub-batch experiments*

Tables C.11 and C.13 show the action times computed from the timing tables. As expected, the total assembly time for six products (975 s) is smaller than six times the assembly of a single product from kit with the TES mounted gripper strategy ($6 * 181 = 1080$ s). One idea behind the sub-

Figure C.1: Pallet Flow in the ASE site using the sub-batch principle

Pallet number	Station number	Remark
2	3	First fixture pallet ready to receive the side plates in the product position
3	3a	Second fixture pallet waiting for station 3 (the product position) to get free.
1	4	Third fixture pallet circulating in the pallet buffer area, waiting for its turn.
5	5	Part pallet with six side plates waiting for its part to be assembled at the part position
8	2	Part pallet with pegs, waiting at the manual station to be send to the part position.
14	2a	Part pallet with shafts, handles and spacer blocks waiting in the buffer of the manual station for its turn to be send to the part position.

Table C.9: *Description of the stations in the VTS used for the sub-batch experiments*

batch strategy is effective: the effect of gripper exchange time is reduced.

Action	Time	Remark
Start exchange gripper 1	16.34.38	Robot empty, above rack
Start picking up side plate 1	35.00	Timed when part was grasped
End mounting side plate 2	35.17	Product pallet briefly jammed
Start picking up side plate 3	35.35	
End mounting side plate 4	36.02	
Start picking up side plate 5	36.31	
End mounting side plate 6	36.59	Change gripper and part pallet
End pickup gripper 2	37.40	
New pallet (no. 8) available	38.09	Transport Command Mixup (TCM)
Start picking up peg 1	38.09	
End mounting peg 8	39.37	
Start picking up peg 9	39.49	
End mounting peg 16	41.12	
Start picking up peg 17	41.24	
End mounting peg 24	42.51	Change gripper and part pallet
End pickup gripper 3	43.31	
New pallet (no. 14) available	44.19	TCM: Made two extra rounds
Start picking up shaft 1	44.19	Three part types
End mounting Block (Sppce) 2	45.39	Misalignment assembly Block
Start picking up shaft 3	45.52	
End mounting Block (Sppce) 4	47.13	
Start picking up shaft 5	47.25	
End mounting Block (Sppce) 6	48.48	
End mounting peg 24	42.51	Change gripper and part pallet
End pickup gripper 1	49.24	
New pallet (no. 5) available	49.21	Pallet was refilled manually
Start picking up side plate 1	49.24	
End mounting side plate 2	49.53	
Start picking up side plate 3	50.03	
End mounting side plate 4	50.31	
Start picking up side plate 5	50.47	Product pallet briefly jammed
End mounting side plate 6	51.15	

Table C.10: Timing of experiment 5: Sub-batch assembly with TES based grippers and the robot speed at 20%

Action	Time	Remark
First gripper exchange	22 s	Includes rack to part pallet motion
First 2 Side plates	17 s	Timing started about 8 s too late
New product from buffer	18 s	Product pallet briefly jammed in corner
Second 2 Side plates	27 s	
New product from buffer	29 s	Transport command mixup (TCM)
Third 2 Side plates	28 s	
Tool exchange time	41 s	
Part pallet change time	70 s	TCM: part pallet makes extra tour
First 8 pegs	88 s	
New product from buffer	12 s	
Second 8 pegs	83 s	
New product from buffer	12 s	
Third 8 pegs	87 s	
Tool exchange time	40 s	
Part pallet change time	89 s	TCM: part pallet makes extra tours
First 2 * 3 pieces	80 s	
New product from buffer	13 s	
Second 2 * 3 pieces	81 s	
New product from buffer	12 s	
Third 2 * 3 pieces	83 s	
Tool exchange time	36 s	
Part pallet change time	33 s	Note: Tool exchange time is critical
First 2 Side plates	29 s	
New product from buffer	10 s	
Second 2 Side plates	28 s	
New product from buffer	16 s	TCM, product pallet jammed briefly
Third 2 Side plates	28 s	
Total assembly time	975 s	Excl. first tool change
Tool exchange time	117 s	Overlaps part pallet exchange
Part pallet change time	162 s	Of which 63 s unnecessary delay
Prod. pal. change time	122 s	
Rack to part pallet	15 s	Estimated overhead time
Net assembly time	663 s	111 s per product

Table C.11: Computed action times of experiment 5

Action	Time	Remark
Start: Moving to first part	14.57.30	Robot w. gripper above rack
Start picking up side plate 1	57.36	
End mounting side plate 2	57.48	
Start picking up side plate 3	58.02	
End mounting side plate 4	58.17	
Start picking up side plate 5	58.50	
End mounting side plate 6	59.05	Change gripper and part pallet
End pickup gripper 2	59.22	
Part pallet (no. 8) available	59.33	Transport command mixup (TCM)
Start picking up peg 1	59.33	First 5 s rack to part pallet
End mounting peg 8	15.00.32	
Start picking up peg 9	00.43	
End mounting peg 16	01.37	
Start picking up peg 17	01.47	
End mounting peg 24	02.40	Change gripper and part pallet
End pickup gripper 3	02.58	
Part pallet (no. 14) available	03.41	TCM: Pallet made extra tour
Start picking up shaft 1	03.46	
End mounting Block (Sppce) 2	04.42	Product pallet briefly jammed
Start picking up shaft 3	04.58	
End mounting Block (Sppce) 4	05.57	Product pallet briefly jammed
Start picking up shaft 5	06.11	
End mounting Block (Sppce) 6	07.13	Change gripper and part pallet
End pickup gripper 1	07.31	
Part pallet available	08.30	TCM, pallet manually refilled
Start picking up side plate 1	08.35	
End mounting side plate 2	08.48	
Start picking up side plate 3	08.56	
End mounting side plate 4	09.12	
Start picking up side plate 5	09.19	
End mounting side plate 6	09.35	
End restore gripper	09.46	

Table C.12: Timing of experiment 6: Sub-batch assembly with TES based grippers and the robot speed at 100%

Action	Time	Remark
First 2 Side plates	12 s	Product pallet briefly jammed Transport command mixup (TCM)
New product from buffer	14 s	
Second 2 Side plates	15 s	
New product from buffer	33 s	
Third 2 Side plates	15 s	
Tool change time	17 s	
Part pallet change time	28 s	
First 8 pegs	54 s	5 s rack to pallet move correction
New product from buffer	11 s	
Second 8 pegs	54 s	
New product from buffer	10 s	
Third 8 pegs	53 s	
Tool change time	18 s	TCM: part pallet makes extra tours
Part pallet change time	61 s	
First 2*3 pieces	56 s	Product pallet jammed briefly Product pallet jammed briefly
New product from buffer	16 s	
Second 2*3 pieces	59 s	
New product from buffer	14 s	
Third 2*3 pieces	62 s	
Tool change time	18 s	TCM: part pallet makes extra tours
Part pallet change time	59 s	
First 2 Side plates	13 s	
New product from buffer	8 s	
Second 2 Side plates	16 s	
New product from buffer	7 s	
Third 2 Side plates	16 s	
Total assembly time	725 s	Overlaps part pallet change time Estimated overhead 74 s per product
Tool exchange time	53 s	
Rack to pallet motion	12 s	
Part pallet change time	148 s	
Prod. pal. change time	123 s	
Net assembly time	442 s	

Table C.13: Computed action times of experiment 6

Appendix D

Implemented primitives

Planning Primitive	Estimated cost
select_robot	0
select_grip	0
select_sensor	0
Macro Primitive	
part_feed	1
part_grasp	3
part_mount	3
extended_line_scan()	2
vision_three_point_measure	6
change_tool	4
Low Level Primitive	
robot_move	2
robot_position	0
sensor_measure	3
gws_grasp	2
gws_release	2
gripper_grasp	1
gripper_release	1
flip_up();	1
flip_down();	1
flip_vacuum_on();	1
flip_vacuum_off();	1