Stellingen bij het proefschrift 'Cost Information Tools for Designers' van ir. Leo S. Wierda

Stelling 1
Vanuit algemeen maatschappelijk oogpunt is de vraag of een produkt überhaupt wel gemaakt zou moeten worden belangrijker dan de vraag hoe de kosten verbonden aan het ontwikkelen en fabriceren van dat produkt geminimaliseerd kunnen worden. De invulling van het begrip 'winst' die nu gebruikt wordt om de eerste vraag te beantwoorden zal drastisch moeten worden herzien.

(From a general social point of view the question whether a product should be made in the first place is more important than the question how the costs associated with the development and manufacture of this product can be minimized. The concept of 'profit' that is now being used to answer the first question will have to be revised drastically.)

Stelling 2
Het modelleren met behulp van features, en op features gebaseerde produktmodellen zullen een hoofdrol spelen in de volgende generatie CAD-systemen.

(Feature modelling and feature-based product models will play a major role in the next generation of CAD-systems.)

Stelling 3
Hoewel het om verschillende redenen aantrekkelijk lijkt om kosten te specificeren per ontwerp-feature, is dit in de meeste gevallen af te raden.

(Although it seems attractive for several reasons to specify costs per design-feature, this is to be discouraged in most cases.)

Stelling 4
Het bouwen van een expert systeem brengt het risico met zich mee dat niet het probleem zelf wordt bestudeerd, maar de wijze waarop de mens dit probleem nu oplost.

(Building an expert system entails the risk that not the problem itself is studied, but the way in which human beings now solve the problem.)
Stelling 5
Er is geen enkel terrein waarop de woorden de daden zo overtreffen als dat van de Kunstmatige Intelligentie.

(Original: 'There is no field in which the claims so exceed the accomplishments as Artificial Intelligence', by FR Brooks as quoted by D. Parnas in 'Why engineers should not use Artificial Intelligence', INFOR, 26, 4, 1988, p. 234.)

Stelling 6
Vergeleken bij het ontwikkelen van kosten informatie systemen voor ontwerpers is het ontwikkelen van eindige elementen programma's voor sterkte- en stijfheidsanalyse een eenvoudig probleem.

(Compared to the development of cost information tools for designers, the development of finite element programs for stiffness and strength analysis is a simple problem.)

Stelling 7
Een vergelijking van de huidige betekenis van het woord 'privé' met die van zijn stam, het latijnse 'privare' (ontnemen), onthult een belangrijke verschuiving in sociale waarden.

(A comparison of the current meaning of the word 'private' with that of its root, the latin 'privare' (to deprive), reveals an important shift in social values.)

Stelling 8
De tweede wet van Murphy vormt een waardevolle aanvulling op de statistiek.

(The second law of Murphy is a valuable supplement to statistics.)

Stelling 9
Gezien de uitermate geringe functionaliteit van stropdassen is het verwonderlijk dat zoveel mannen dagelijks aan de galg willen worden herinnerd.

(Considering the very limited functionality of neckties it is surprising that so many men want to be reminded daily of the gallows.)

Stelling 10
De geur van jonge lentebloesem op een toilet is een voorbeeld van geaccepteerde luchtvuiling.

(The smell of young spring blossom on a toilet is an example of accepted air pollution.)
Cost Information Tools for Designers

(a survey of problems and possibilities with an emphasis on mass produced sheet metal parts)

Kosten Informatie Systemen voor Ontwerpers

(een overzicht van problemen en mogelijkheden met speciale aandacht voor in massa geproduceerde plaatwerkdelen)

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 13 september 1990 om 14.00 uur

door Leo Simon Wierda,

Werktuigbouwkundig Ingenieur,
geboren te Schagerbrug op 6 juli 1957
Dit proefschrift is goedgekeurd door de promotoren:

Prof. dr. ir. H. Hermans (Technische Universiteit Delft)

en

Prof. Dr.-Ing. G. Warnecke (Universiteit Kaiserslautern)

CIP-gegevens Koninklijke Bibliotheek, Den Haag

Wierda, Leo Simon

Cost information tools for designers: (a survey of problems and possibilities with an emphasis on mass produced sheet metal parts) / Leo Simon Wierda. - Delft : Delftse Universitaire Pers. - Ill.

Proefschrift Delft. - Met lit. oog. - Met samenvatting in het Nederlands

ISBN 90-6275-592-5

SISO 655.1 UDC 681.3:338.512:669.141(043.3) NUGI 862

Trefw. : kosteninformatiesystemen / plaatwerkdelen

© Copyright 1990, Delft University Press.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission, in writing, from the publisher. For permission contact D.U.P., Stevinweg 1, 2628 CN. Delft, Netherlands

Printed by: Universiteitsdrukkerij T.U.-Delft
Contents

1 Introduction
1.1 The potential influence of a designer on product costs
1.2 Towards structural attention for cost reduction
1.3 Problems in effective cost control
1.4 The need for new tools
1.5 Research objective

2 Cost information in the product design process
2.1 Types of costs
   2.1.1 Non-financial costs
   2.1.2 Financial costs
2.2 The Basic Cycle in the design process
2.3 The three phase model for the design process
   2.3.1 Conceptual Design
   2.3.2 Materialization (or Design)
   2.3.3 Detailed Design
2.4 Types of Cost Information
   2.4.1 Cost structures
   2.4.2 Design rules and Good/Bad examples
   2.4.3 Relative Costs
   2.4.4 Global Cost Estimation Methods
   2.4.5 Detailed Cost Estimation
2.5 Some problems associated with Cost Information
   2.5.1 Conceptual Designs
   2.5.2 Required organizational measures
   2.5.3 Cost Information is not always economical
   2.5.4 Time dependency of the information
   2.5.5 Company dependency of information
   2.5.6 Accuracy of global cost estimation methods
2.6 Summary

3 Tools and techniques
3.1 Qualitative Cost Information tools
   3.1.1 Passive approach
   3.1.2 Some active approaches
3.2 Global Quantitative Cost Information
   3.2.1 Cost Functions
   3.2.2 Derivation and use of Cost Functions
   3.2.3 The KOINF program by Kreisfeld
   3.2.4 Other programs for global cost information
3.3 Detailed Quantitative Cost Information
   3.3.1 Derivation and use of detailed cost information
   3.3.2 Ferreirinha: the HKB-program
   3.3.3 Programs by Baumann
   3.3.4 Other programs for detailed cost estimation
3.4 The KIS approach by Ehrenspiel and Hillebrand
3.5 Summary
4 Outlines for the DIDACOE programs

4.1 Introduction
4.2 Mass production
4.3 Cost estimation
4.4 Process planning
4.5 The use of features
4.6 Manufacturability information
4.7 Neutral file formats
4.8 A class of sheet metal parts
4.9 Survey

5 Detailed cost estimation concept

5.1 Main Objects and Product Structure
5.2 Operations and additional objects in the Production Path
5.3 A black box model for operations
5.4 Quantities
  5.4.1 Total Quantity and Batch Quantity for Main Objects
  5.4.2 Quantities in a Production Path
5.5 Types of costs
5.6 Estimation of material costs
  5.6.1 Introduction
  5.6.2 Costs of Basic Objects, BuyParts and Added Material Objects
  5.6.3 Estimation of MakePart material costs
  5.6.4 Estimation of Product material costs
5.7 The grey box model for operations
  5.7.1 Actions and Times
  5.7.2 Machines, Operators and Rates
5.8 Estimation of operation costs
  5.8.1 Determination of operation times
  5.8.2 Determination of machine and operator rates
  5.8.3 Operation costs for a MakePart
5.9 Estimation of tool costs
5.10 Information Summary

6 Cost estimation for a class of sheet metal manufacture

6.1 General description of the type of operation
6.2 Operation type model
  6.2.1 Input and Output
  6.2.2 Actions
  6.2.3 Machine
  6.2.4 Operators
  6.2.5 Tool
6.3 Strip LayOut
6.4 Number of Strokes and Number of Parts per Strip
6.5 Action Times
  6.5.1 Time for the main action
  6.5.2 Times for secondary actions
  6.5.3 Time for preparative actions
  6.5.4 Determination of the cycle time
6.6 Press selection procedure
6.7 Tool cost estimation
6.8 Information Summary

7 Basic Feature Description and Recognition
7.1 Introduction
7.2 Feature descriptions
7.2.1 Compound basic features
7.2.2 Elementary basic features
7.2.3 Bends, bend lines and faces
7.2.4 Special conditions and tolerances
7.3 Feature Recognition Program
7.3.1 Input: SMD and the corrected development
7.3.2 Output: NXP-database-format files
7.3.3 Program outlines
7.4 Evaluation

8 Strip Layout generation and Advanced Features
8.1 Introduction
8.2 Existing programs for layout design
8.3 Outlines for the new strip layout generation program
8.3.1 General program philosophy
8.3.2 The main output: Strip Layout description
8.3.3 Additional output: Manufacturability Information
8.4 Influence of Outer Contour shape on layout design
8.4.1 General principles for cut off
8.4.2 Special cut off situations
8.4.3 Special cut out situations
8.5 Influence of bend lines on layout design
8.5.1 Bend lines and grain direction
8.5.2 Special layouts for bending
8.6 Other influences on layout design
8.6.1 Corner Angles
8.6.2 Tolerances and edge conditions
8.6.3 Other considerations
8.7 Edge Models and Advanced Features
8.7.1 Introduction
8.7.2 Groups of LIN, Edge Holes and Straight Edges
8.7.3 Groups of RAD, Circle Interruptions and Circle Edges
8.7.4 Straight Edge Model creation without suitable AGL
8.7.5 Straight Edge Model creation with suitable AGL
8.8 Edge Relations
8.8.1 Uniformity
8.8.2 Angles and intersection lengths
8.9 Layout creation
8.9.1 Layouts for cut off
8.9.2 Layouts for cut out
8.9.3 Pitch determination for non-trivial cases
8.10 Program evaluation
8.11 Notes on the development of a tool layout design program
9 The Knowledge-Based System experience
  9.1 A short introduction to Artificial Intelligence
     9.1.1 What is AI?
     9.1.2 The main AI-techniques
     9.1.3 What is a KBS?
     9.1.4 When use AI-techniques?
  9.2 AI-techniques in DIDACOE
  9.3 Selection of an environment for KBS-developement
  9.4 Nexpert Object
  9.5 Strip Layout Design as a KBS
  9.6 Nexpert evaluation

10 Global Process Planning
  10.1 Introduction
  10.2 State of the art in CAPP
     10.2.1 Variant versus Generative systems
     10.2.2 Use of AI-techniques
     10.2.3 Progress in problem solution
  10.3 Elements in sheet metal manufacture
     10.3.1 Types of Basic Objects
     10.3.2 Types of Operations
     10.3.3 Types of Machines
     10.3.4 Types of Tools
  10.4 Possibilities for CAPP in sheet metal manufacture
  10.5 Generation of manufacturability information

11 Data Storage
  11.1 Runtime, temporary and permanent storage
  11.2 Selection of data for storage
  11.3 Data structures

12 Conclusions and Recommendations

Appendix 1 General data and datastructures
Appendix 2 Sheet Metal specific data and datastructures

List of symbols and abbreviations
List of references
Curriculum vitae
Summary

This thesis discusses the possibilities to develop computerized cost information tools for designers. Associated problems that receive attention are: the use of feature models, automated process planning, structures for storage of the associated data, and the use of artificial intelligence techniques. More in specific the work focuses on cost estimation for mass production of sheet metal parts from strip in progressive dies.

The first three chapters introduce the cost information problem. The current possibilities to minimize manufacturing costs during design are clarified, and the need for new cost information tools is explained. A global analysis of the design process provides insight in the types of information that a designer needs. This includes both qualitative and quantitative information. The tools and techniques that are required to supply it are addressed in chapter 3. This chapter also presents a survey of currently available tools.

Chapter 4 presents a general concept for a cost information tool for designers, called DIDACOE. It also explains the limited implementation of this concept for sheet metal parts (see figure 4.6). This implementation provides detailed quantitative cost information only.

A concept for detailed product cost estimation is presented in chapter 5. This general concept can be extended with operation specific concepts for the estimation of operation times and tool costs. An example for the production of sheet metal parts from strip in progressive dies is explained in chapter 6. It only deals with time estimation, but possibilities for tool cost estimation have been indicated.

The detailed cost estimation system requires detailed process planning data as input. As these data can usually not be supplied by the designer, attention was given to automation of the process planning task. A concept for automatic generation of strip layouts has been developed. It aims at finding advantageous layouts by reasoning, using an advanced feature description of the outer contour of unfolded sheet metal parts. This is explained in detail in chapter 8. The concept has been implemented using Artificial Intelligence techniques. Although it is still a prototype, the approach seems promising. It offers possibilities for extension to tool layout generation and to the generation of detailed manufacturability information. More general remarks on process planning can be found in chapter 10.

The advanced feature model for the outer contour is derived from a basic feature description for sheet metal parts. Currently this description is limited to flat parts and to parts with straight bends. A feature recognition program has been developed for automatic generation of the basic feature description from a 2D representation of the parts in the CAD-system Medusa (chapter 7).
Samenvatting

Dit proefschrift behandelt de mogelijkheden om geautomatiseerde kosteninformatiesystemen voor ontwerpers te ontwikkelen. Gerelateerde problemen die aandacht krijgen zijn: het gebruik van 'features', automatisering van de werkvoorbereiding, toepassing van kunstmatige intelligentie technieken en het ontwikkelen van datastructuren. Meer specifiek wordt ingegaan op de massaproductie van plaatwerkprodukten uit stroken via gebruik van volgstempels.

De eerste drie hoofdstukken vormen een inleiding op het kosteninformatie probleem. De aanwezige mogelijkheden om fabricagekosten tijdens het ontwerpen te minimaliseren worden verduidelijkt en de behoefte aan nieuwe kosteninformatiesystemen wordt uitgelegd. Een globale analyse van het ontwerpproces geeft inzicht in de soorten informatie die de ontwerper nodig heeft. Dit omvat zowel kwalitatieve als kwantitatieve informatie. De middelen die nodig zijn om deze informatie te verschaffen komen in hoofdstuk 3 aan de orde. Dit hoofdstuk geeft ook een overzicht van de uit beschikbare systemen.

In hoofdstuk 4 wordt een algemeen concept voor een kosteninformatiesysteem gepresenteerd, onder de naam D-IDACOE. Hier wordt ook de beperkte implementatie van dit concept voor plaatwerk geschetst (zie figuur 4.6). Dit programma verschaf alleen gedetailleerde kwantitatieve informatie.

Een concept voor gedetailleerde produktkostenschatting wordt in hoofdstuk 5 beschreven. Dit algemene concept kan worden uitgebreid met bewerkingsspecifieke concepten voor de schatting van bewerkingsstijden en gereedschapskosten. Een voorbeeld voor de produktie van plaatwerkprodukten uit stroken in volgstempels is in hoofdstuk 6 te vinden. Het betreft hier alleen tijdschatting, maar de mogelijkheden voor gereedschapskostenschatting worden wel aangeduid.

Het kostenschattingssysteem heeft gedetailleerde bewerkingsgegevens als invoer nodig. Omdat de ontwerper deze gegevens doorgaans niet kan verstrekken, is aandacht besteed aan het automatiseren van de werkvoorbereiding. Dit heeft een concept voor automatische generatie van strookindelingen opgeleverd. Het doel van dit concept is om gunstige indelingen te vinden via een redeneringsproces. Daartoe wordt de buitenomtrek van de uitslag van een plaatwerkprodukt beschreven via 'advanced features' (hoofdstuk 8). Het concept is geïmplementeerd met behulp van technieken uit de kunstmatige intelligentie. Hoewel het nog een prototype betreft, ziet de benadering er interessant uit. Het biedt mogelijkheden voor uitbreiding tot het genereren van gereedschapsindelingen en van gedetailleerde bewerkbaarheidsinformatie. Hoofdstuk 10 geeft algemene opmerkingen over het automatiseren van de werkvoorbereiding.

De 'advanced features' worden afgeleid uit een beschrijving van het werkstuk via basisfeatures. Deze beschrijvingsmogelijkheid is nu beperkt tot vlakke werkstukken en tot werkstukken met rechte buigingen. Een feature-herkenningsprogramma is ontwikkeld om de beschrijving via basisfeatures automatisch te kunnen genereren op grond van een 2D representatie van de plaatwerkdelen in het CAD-systeem Medusa (hoofdstuk 7).
1 Introduction

1.1 The potential influence of a designer on product costs

In my industrial experience, which now spans 35 postcollege years, the most significant manufacturing cost reductions and cost avoidances are those that result from changes in product design rather than from changes in manufacturing methods or systems. With this opening statement in his preface to the Handbook of Product Design for Manufacturing /Bra-1/, James G. Bralla emphasizes the large potential influence of designers on product costs.

This influence is also demonstrated quite clearly by examples of cost reductions that were obtained using Value Analysis and Engineering /Mil-1/. Although this technique does not only consider the design itself, but purchasing and manufacturing aspects as well, it is apparent from the examples that design decisions play a major role.

The results of German research /Vdi-1,Ehr-2,Rau-1/ allow a quantitative illustration of the potential influence of a designer on product costs. This research (limited to machine design) shows that, although the costs of the design department itself (mainly wages) constitute only about ten percent of the eventual product costs, this department fixes 70 to 80% of these costs (figure 1.1). Kiewert pointed out that this percentage is misleading. The product specification from which the designer starts his efforts, already implies some minimal costs that can hardly be influenced /Kie-1/. It is better therefore, to

![Diagram showing the influence of different company departments on product costs](image)

**Figure 1.1:** Influence of the main departments in a company on the costs of a product according to German research. Each department causes costs itself but also makes decisions that cause costs in other departments. That is: it also fixes costs. Note that although the design department causes only 10% of product costs, it fixes about 70% of these costs. (Source: /Vdi-1/, adapted)
state that the design department can be held responsible for two thirds of the unnecessary costs: 20 to 30 percent of the total product costs. It was Ehrlenzpiel who derived these figures by comparing product costs before and after Value Analysis and by noting whether changes were made in the design, in the process planning or in the purchase of material /Ehr-3/.

1.2 Towards structural attention for cost reduction

Good, cost effective designs are not new. In Bralla’s words: “Since the dawn of mass production, a century and a half ago, innumerable products have been developed which are models of simplicity from a manufacturing standpoint” / Bra-1/. What has been lacking however is a structural attention for cost reduction during the design process.

As illustrated by Tassinari /Tas-1/ two historical developments can be distinguished.

The first thing to note is that early attempts to reduce costs were often limited to automation and more efficient organization of the production process. Relatively little attention was paid to the design itself. Probably, this changed when the application of Value Analysis and Engineering /Mil-1/ became more widespread. Although Miles introduced this technique in the forties, and although it was certainly being used in the fifties, experts like Ehrlenzpiel from West Germany and Ostwald from the United States confirm that a more widespread application probably stems from the early sixties, especially in Europe. Apart from this, the work of Kesselring /Kes-1,2/ contributed to an increased attention for cost effective design, especially in German speaking countries.

A second change to note is the one from Value Analysis, which was originally aimed at critical examination of existing products, to a structural attention for cost reduction during the design process of new products. Considering the often surprisingly large cost reductions obtained by Value Analysis, this shift is natural and understandable: try to do it better the first time instead of changing afterwards. Associated with this shift is the ‘Design-to-Cost’-principle: next to other product specifications the client also specifies target costs, thus forcing the designer to work cost efficiently from the start of the design process.

Although the developments described above may not have a general validity, they indicate a major change for the design department. This department is increasingly held responsible for product costs. Considering the potential influence of designers on these costs this seems to be justified. It is not fair however to hold designers responsible if they don’t have the means to make the cost effective decisions that are necessary!
1.3 Problems in effective cost control

Even today, a structural and organized cost control during the design process often lacks. Trying to explain this situation, Ehr's /Ehr-4/ points out that cost reductions should now typically be obtained using the Long Cost Control Circuit, illustrated in figure 1.2. In this circuit, the design department does not perform the cost analysis itself. This task is left to the process planning and cost estimating departments. Information from these departments has to flow back to the design department to create an iterative process in which product costs can be reduced /Ehr-1/. Two factors prevent the execution of this cost control process or limit its efficiency: lack of time and communication problems.

The communication problems are the direct result of the division of labour /Ehr-1/Ehr-5/. All departments have their own way of thinking and their own terminology. In addition the departments tend to protect their position and their

Short Cost Control Circuit       Long Cost Control Circuit

Figure 1.2: The division of labour lead to several departments, each with their own specialized task. This implies that the design made in the design department is not analysed here, but in the process planning and cost estimating departments. An iterative process with feedback of information to the design department is necessary to reduce costs. Lack of time and problems in the communication between the departments limit the efficiency of this Long Cost Control Circuit. Hence the wish to create the Short Control Circuit: cost control by the design department itself. (Source:/Ehr-4/, adapted).
monopoly: they try to turn the situation to their own advantage. This creates 'mental walls' between the departments that hinder the flow of information (figure 1.3). Hence cost and manufacturability information does not always arrive in the design department as it should have. As a result, designers have insufficient knowledge on production, purchase and costs to enable them to make cost effective and production oriented decisions /Ehr-1/.

The lack of time is a problem of course that will arise especially in situations where small quantities of products are to be made. In these cases cost reduction attempts may not be economical due to the fact that the time spent in these attempts costs more than the reduction obtained. Time pressure is becoming a problem everywhere however due to the shortening of product life cycles, the demand for shorter throughput times, the increase in the number of product variants caused by the more client oriented approach of the present, and the increase in the complexity of the design process.

Ehrlenspiel proposes the Short Cost Control Circuit to solve these problems: cost analysis of the design inside the design department.

Some will argue that Ehrlenspiel's cost control model is too simple, too theoretical or not generally valid. His approach shows the essential problem quite clearly however: the need for feedback of information from process planning, cost estimating and production to the design department. The lack of time, the amount and the complexity of the information, combined with the communication problems described above hamper this information feedback. In addition it should be noted that the design task and the process planning/production tasks are not always performed within the same company. In these cases the feedback problem may even be larger.
1.4 The need for new tools

On the one hand, organizational measures can improve the situation described above. On the other hand new tools can be developed that provide the designer with the qualitative and quantitative information necessary to make cost effective designs. Although a combination of both will be required, the organizational measures fall beyond the scope of this research. This thesis focuses on the development of new tools.

Making cost information and cost estimation methods available to the design department does not imply that this department has to perform the tasks currently handled by the process planning and cost estimating departments. The designers neither have the experience nor the time to do this work. The last two departments should prepare the cost information in such a way that it is suitable for use by the designer. In the first place this implies that the information has to be related to the decisions that the designer has to make. In addition, time-consuming search for data and tedious calculations should not be necessary. The information should be easy and fast to access, it should be reliable and up to date, and it should not only tell the designer how much a design decision will cost, but also how the design should be adapted to make it cheaper.

This character of the cost information and the position of the designer imply that we can not just transport the methods and data now available in the process planning, purchasing and cost estimating departments to the design department. Data have to be prepared and presented in a suitable way to be useful for the designer. New tools are necessary, both to prepare the information for the designer and to make this information available to the designer.

Handbooks like the Handbook of Product Design for Manufacturing /Bra-1/ or the AM Cost Estimator /Ost-1/ are one means of presentation. For several reasons however a computerized tool offers advantages. First it will be easier to keep all information up-to-date if it is stored in a database in the computer. Second, computer programs will be necessary anyway to facilitate cost calculations. Third, Computer Aided Design systems (CAD-systems) are quite commonly used already. It is to be expected that this use will increase in future. What is more logical therefore than to create a Cost Information System as an integral part of the CAD-environment in which the designer works. The additional advantage of such an integration is that the Cost Information System can take the product data from the internal product representation defined by the CAD-system, if necessary. Thus we avoid the need to specify product data more than once.

In this way the cost information tool can be compared with other information or analysis tools linked to CAD, such as finite element analysis or production simulation programs. As such, it may contribute to the Computer Integrated Manufacturing (CIM) concept.
1.5 Research objective

The original objective of the research project described in this thesis was to improve and extend existing programs for detailed cost estimation of sheet metal parts and to link them to the CAD-system Medusa. It was soon recognized however that the development of these programs should be the first step on the way to a more general cost information tool for designers. Therefore a more general second objective was added: to clarify the possibilities for the development of computerized cost information tools for designers and to properly understand all the problems involved.

This second objective is most clearly reflected by the contents of chapters 2 and 3: they consider cost information tools for designers in general. Chapter 2 explains the types of cost information that a designer might need during the design process. Chapter 3 explores how these types of information can be generated and also provides a survey of already available tools. It distinguishes four types of tools: database search for qualitative cost information, active generation of manufacturability information, tools for global cost estimation and tools for detailed cost estimation.

All following chapters consider detailed cost estimation and active generation of manufacturability information only. The second objective was more specifically pursued for these two types of tools. Chapter 4 presents a general concept for a cost information system based on these two types of tools and motivates this choice. In addition it shows which part of this general concept has actually been implemented. This implementation reflects the original objective.

The theoretical concepts on which the implemented programs are based are explained in the following four chapters. Chapter 5 describes the concept for the central detailed cost estimation program. Amongst others, this program requires operation times and basic tool costs as input. This input can be supplied by operation type specific modules. A concept for one such module, for the production of sheet metal parts from strip in progressive dies, is explained in chapter 6. In its turn, this module also requires input. Most important input data are the strip layout and the tool layout. Chapters 7 and 8 describe how these layouts can be derived automatically using a feature description for sheet metal parts that is obtained by interpretation of CAD-data on these parts.

The remaining chapters (9, 10 and 11) deal with some special topics.
2 Cost information in the product design process

2.1 Types of costs

2.1.1 Non-financial costs

Every product fulfils some function. In doing so, and by its mere existence, the product influences the life of individual human beings and the life in society as a whole. These influences can be both positive and negative. If they are negative they could be conceived as psychological or sociological costs. Although a lot can be said about the impact of products like cars, televisions or telephones on the everyday life, their influences can hardly be separated from influences caused by other products. In addition it seems almost impossible to assess these influences in some quantitative way.

All products together seem to create an ever increasing speed of life (more has to be done, more happens in less time) and an ever decreasing need for physical human efforts. Clearly this development is far beyond the control of the individual designer. The philosophical questions can be raised whether anybody can control this development and whether it needs to be controlled or not.

Somehow related to the above are environmental costs. Every product will have some influence on the natural environment in which we live and on which our life depends, either during its production, during its use or because of its disposal. To some extent these environmental costs are inevitable. Just like all other life forms human beings have to use part of their environment to survive. On the long term we have to ensure however that the environment remains suitable to live in. That is: we should not use or destroy too much too fast. Especially during the last decade the attention for this topic is growing.

Designers have a clear influence on environmental costs. They choose the type and amount of material to be used, fix the type of operations necessary to manufacture the product, determine to a large extent the product emissions and influence the possibilities for recycling of product material. Product emissions are and will increasingly be limited by restrictions imposed by the government. Research is ongoing in the Netherlands (Delft, Leiden) to give the designer information on the 'energy content' and the 'environmental burden' of materials and operations (/Kem-1,Ste-1/). In addition information on recycling oriented design is becoming available /Wee-1/.

Although the topics discussed briefly above are very interesting they fall beyond the scope of this research.
2.1.2 Financial costs

The term 'financial costs' is used here to indicate all costs that are expressed in an amount of money, as opposed to the psychological, sociological and environmental costs mentioned above. When we talk about costs in this report we mean financial costs unless indicated otherwise.

Within the group of financial product costs it is important to distinguish life cycle costs, company costs and manufacturing costs /Vdi-2/.

Life Cycle Costs (german: Produkt Gesamtkosten) are all the costs for the user during the period in which the product is used. This includes once-only costs to buy the product (strictly speaking these are expenditures, not costs), costs for transport and installation, costs for training and instruction and costs for product destruction or disposal. Continuous costs include energy consumption, wages for the human operator and costs for repair and maintenance. The designer has a considerable influence on these costs /Vdi-2,Fra-1/ but this topic will receive little attention here.

Company Costs (german: Selbstkosten) are all the costs a company makes to develop, manufacture and market a product. The costs are often related in some way to the price at which the product sells at the market. The difference between the two is the company profit or loss.

Manufacturing Costs (german: Herstellkosten) are all the costs made to manufacture the product. The manufacturing costs are the sum of material costs and operation costs. In principle they exclude costs for development and design, marketing costs and other comparable company costs.

The term Product Costs in this report will usually refer to manufacturing costs. The other company costs are often assigned to products using one or more allowances on the material, operation or manufacturing costs. Therefore product costs as used in this report may include other elements of the company costs, depending on the allowances defined. This will be clear from the context. Fixed and variable costs, direct and indirect costs, the difference between costs, expenditures and losses and other business economical topics will not be explained here. Comprehensible explanations can be found in /Koo-1,Mal-1,Vdi-1,Vdi-4,Vdi-5,War-1/ and in the general literature on business economics.

2.2 The Basic Cycle in the design process

According to Eekels /Eek-1/ the designer applies what he calls the 'Basic Cycle' to solve problems during the design process (figure 2.1). This cycle consists of:

- Analysis: study the problem to be solved and define it clearly.
- Synthesis: generate solution(s) for the problem.
- Simulation: determine or estimate the properties of the solution.
- Evaluation: compare the properties of the solution(s) with the desired properties defined during the Analysis. Depending on the outcome accept the solution, adapt the problem definition or generate other solutions.
The Basic Cycle is not only applied to solve the total design problem, but for the solution of all partial problems as well, although maybe not consciously.

From the point of view of cost information three points should be noted:
- cost estimation (the determination of the costs of a design or of a part thereof) belongs to the Simulation phase of the Basic Cycle,
- the results of a cost estimate have to be used by the designer during the Evaluation phase. This implies that the information provided by the estimate should be such that it can be compared with the demands defined during the Analysis phase. In addition the information should indicate how the (partial) design has to be adapted to improve its cost properties: cost information should be related to parameters that can be influenced by the designer,
- the designer may need cost information to be able to define the desired cost properties in the Analysis phase. In other words: he or she needs some reference information to know what cost properties to expect.

![Diagram](phase in Design Process)

<table>
<thead>
<tr>
<th>Phase in Design Process</th>
<th>Cost related task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>Define target costs</td>
</tr>
<tr>
<td></td>
<td>Determine main factors</td>
</tr>
<tr>
<td></td>
<td>that influence costs</td>
</tr>
<tr>
<td>Synthesis</td>
<td>Estimate costs</td>
</tr>
<tr>
<td></td>
<td>(determine value)</td>
</tr>
<tr>
<td>Simulation</td>
<td>Compare target costs</td>
</tr>
<tr>
<td></td>
<td>with estimated costs</td>
</tr>
</tbody>
</table>

**Figure 2.1**: The Basic Cycle for the Design Process according to Eekels /Eek-1/. The determination of target costs and main cost influencing factors is part of the Analysis. Estimation of costs belongs to the Simulation. Demands defined during Analysis are compared to estimates during the Evaluation. To get an effective iterative process, cost information should be related to factors that can be influenced by the designer.
2.3 The three phase model for the design process

In Germany design methodology has provided a three phase model for the design process, which has been accepted by the German Association of Engineers /VDi-3/ and is also adopted by Eekels at the University of Delft in the Netherlands /Eek-1/ This model distinguishes three main phases in the design process:
- Conceptual Design (German: Konzipieren)
- Materialization or Design (German: Entwerfen)
- Detailed Design (German: Ausarbeiten)

The model is particularly interesting as it has been used by the German Association of Engineers to explain the types of cost information that a designer will need during the design process (Figure 2.2, /VDi-2/).

Figure 2.2: Main tasks for the designer in the three phase model for the design process and associated types of cost information. The link between design tasks and types of information is only an example. (Source: /VDi-2/, simplified)
2.3.1 Conceptual Design

The assignment given to the designer marks the startpoint for the conceptual phase of design. From the point of view of cost effective design it is of prime importance that this assignment includes a specification of target costs /Ehr-1, Vdi-2, Tan-1/.

The first task for the designer is to study the assignment and to define the design problem clearly and in more detail. The designer should try to determine which factors or which parts of the product will have the main influence on costs. These factors or parts should get special attention during design. To be able to perform this analysis the designer will need reference cost information on existing products that are similar to the product to be designed. Especially cost structures and ABC-analysis /Vdi-2/ are helpful here.

When the list of demands is clearly defined the designer has to:
- identify subfunctions within the total product function,
- search basic solutions for these subfunctions. Design catalogues /Vdi-3/, maybe including relative function costs, and general design rules can be helpful here,
- combine potential partial solutions to conceptual product designs,
- choose the most promising conceptual design(s) (one or a few) as the base for further work. Methods for design assessment /Vdi-4, Pah-4/ that take into account both technological and economical factors are necessary here. Possibly, the design assessment is performed by a group of specialists from different departments: the so called conference method /Ost-2, Vdi-2/.

2.3.2 Materialization (or Design)

The name for this phase, Materialization, has been taken from Eekels /Eek-1/. Ostwald calls it the Design phase /Ost-2/, similar to the german name 'Entwurfsphase' /Vdi-2/.

During this stage the types of materials to be used are determined and the product is dimensioned. Although the materialized design is a design of the complete product, some critical features may still be open for later detailing. A sketch or a drawing of the design is available /Eek-1/.

Types of cost information that will be necessary here are:
- cost structures, to analyse which factors or parts have the highest influence on costs,
- relative costs for materials, form features, types of operations, types of fasteners, types of bearings, tolerances, surface qualities, and the like, to speed search for the cheapest solution,
- design rules, maybe presented graphically in the form of good/bad construction examples,
- catalogues on standard buy-parts, on make-parts used in other company products ('Wiederholteile') and on standardized dimensions within families of parts, to avoid additional design efforts and to increase the quantities to buy or make,
- break even points for alternative manufacturing processes.
- global cost estimation methods to obtain a fast estimate of the absolute
costs or to compare alternatives.
(See /Vdi-2/ for a more complete and detailed list)

2.3.3 Detailed Design

During the detailed phase of design details and parts of the product are
optimized. Maybe a prototype is built and tested. All drawings, bills of materials
and other documents needed for process planning and manufacturing are
produced during this stage /Eek-1/. Maybe the design of special tools needed
during production and assembly should also be regarded as part of this phase.
All cost information mentioned above will also be helpful in this stage. In
addition methods for detailed cost estimation can be used to get a more
accurate estimate of the product costs, to be compared with the target costs
defined in the assignment.

2.4 Types of Cost Information

2.4.1 Cost structures

A cost structure is a division of costs in several parts. The parts can be
assigned either absolute or relative costs (percentages). Cost structures can
be defined for manufacturing costs or for life cycle costs. In the first case the
parts to be distinguished can be component parts and subassemblies,
subfunctions within the product or types of costs. In the second case types of
costs will be distinguished as indicated in 2.1.2 and in figure 2.3. Examples of
cost structures can be found in /Vdi-2,4/.

Cost structures will be used to find the product components, the product
functions or the types of costs that have the highest influence on the costs.
Hence they indicate to what aspect of the product the designer should direct
most of his attention. Although they can be useful during all phases, cost
structures are particularly important during conceptual design.

2.4.2 Design rules and Good/Bad examples

Design rules are probably the most widely used type of cost information. They
contain a large part of the experience gained in previous design and
manufacturing processes and thus are the base for 'design for manufacturing'.
Many rules can be found in literature. They range from very global to very
detailed.
Examples of global rules are: 'Reduce the number of parts required where
possible by designing one part so that it performs several functions' or
'Whenever possible, design to use general-purpose tooling rather than special
tools'. An example of a more detailed rule could be: 'The diameter of circular
holes in sheet metal parts should be larger than the sheet thickness and
preferably at least 2.5 mm'.
Figure 2.3: Cost structures for Life Cycle Costs of a spanner, a car and a water pump. Cost structures are useful to find the components or subassemblies, the product functions or the types of costs that have a main influence. Hence they point to the main factors that need the designer's attention. (Source:/Vdi-2/, adapted)

Sometimes the rules will not have a general validity and it may not be well defined when it is allowed to apply them.

Design rules can often be presented very clearly by graphical examples of good and bad construction. An example is shown in figure 2.4. Many other examples can be found in /Ehr-2,Bra-1,Bod-1/. Information on the deduction and use of design rules can be found in /Bau-1/ for example.

2.4.3 Relative Costs

Relative costs are costs for an object in proportion to the costs for some reference object. Relative costs can be supplied for different types of objects. Examples include functions /Rad-1,2/, materials /Vdi-4/, operations /Hub-1/, different possibilities to join parts /Bus-1,2,3/, standard parts like bearings /Hub-1/, surface qualities /Ste-2/ and so on. An example is shown in figure 2.5.

The main advantage of relative costs (as opposed to absolute costs) is that they provide a fast and easy way to find the cheapest alternative in a collection of possible solutions, using a minimum amount of figures. In addition relative costs are said to be less subject to change than absolute costs (see paragraph 2.5.4). Both advantages are doubtful.

The main disadvantage is that they cannot be used in cost calculations: when target costs have to be checked, relative costs are of limited value.

A lot of research on relative cost catalogues has been done by Schuppar /Sch-2/. He provides tools to construct and maintain catalogues with relative cost information. Radermacher /Rad-1,2/ continued this work. Positive experiences in working with relative costs are recorded by Schulze /Sch-1/ and Steinwachs /Ste-2/.
**Figure 2.4:** Good/Bad examples for deepdrawing design. These examples can be conceived as graphical presentations of design rules. (Source: /Bra-1/)

<table>
<thead>
<tr>
<th>Material</th>
<th>Relative Costs (St37 = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common construction steel</td>
<td>1.0-1.1</td>
</tr>
<tr>
<td>Case hardened steel</td>
<td>1.1-2.3</td>
</tr>
<tr>
<td>Tempered steel</td>
<td>1.2-2.7</td>
</tr>
<tr>
<td>Heat resisting steel</td>
<td>2.0-2.9</td>
</tr>
<tr>
<td>Nitride steel</td>
<td>2.6</td>
</tr>
<tr>
<td>Non-magnetic steel</td>
<td>4.1</td>
</tr>
<tr>
<td>Stainless steel</td>
<td>3.2-5.8</td>
</tr>
<tr>
<td>High temperature steel</td>
<td>5.1-9.2</td>
</tr>
<tr>
<td>Brass</td>
<td>6.8-8.0</td>
</tr>
<tr>
<td>Electrolytic copper</td>
<td>9.5-10.0</td>
</tr>
<tr>
<td>Copper-Tin alloys</td>
<td>17.3</td>
</tr>
<tr>
<td>Mouldable Copper alloys</td>
<td></td>
</tr>
<tr>
<td>Aluminium (pure)</td>
<td>2.3</td>
</tr>
<tr>
<td>Aluminium alloys</td>
<td>2.9-3.9</td>
</tr>
</tbody>
</table>

**Figure 2.5:** Relative material costs according to VDI design guide 2225. Relative costs provide a fast way to choose the cheapest solution within a group of alternative solutions. Their use is limited however by the fact that they can not be used in calculations. (Source: /Ehr-1/)
2.4.4 Global Cost Estimation Methods

Cost structures, relative costs and design rules can be conceived to provide mainly qualitative cost information. They are used to find the main cost influencing factors or the cheapest solution within a group of alternatives.

In addition to this the designer will also need quantitative information. In other words, he or she will want an answer to questions like: 'How much will my design cost?' or 'How much can I save by changing this?'. Cost estimation methods are necessary to provide this quantitative answer. In this report we will distinguish global and detailed cost estimation.

Global cost information is always presented for some class of objects. The objects can be complete products, assemblies, parts, form features, functional units or operations. Classification of objects can depend on very global characteristics such as the product function (car, watch, pump). On the other hand very restricted classes can be found where the cost information is valid only for objects that are similar to some basic object both in geometry and in way of production.

In all cases the information is derived from known costs for existing objects (at least one) in the class. Typically, analysis of existing data will result in the definition of a limited number of parameters (object characteristics) that have a main influence on costs. Often this leads to some cost function.

An example of such a cost function for pressure vessels (single production), derived by the 'Bemessungslehre' is:

\[
\text{COST} = (1.7*V^{1.1} + 13.8*k_2*R^3 + (0.27*c*V)/R + 12.6*c*R^2) \times (P/(0.8*S))
\]

where \( V \) the vessel volume in \( \text{cm}^3 \), \( R \) the radius of the cylindrical part in \( \text{cm} \), \( c \) a factor related to welding costs in \( \text{DM}/\text{cm}^2 \), \( P \) the maximum pressure in \( \text{bar} \), \( S \) maximum allowable stress in \( \text{N}/\text{cm}^2 \) and \( k_1, k_2 \) specific material costs in \( \text{DM}/\text{cm}^3 \).

Another function has been taken from Diels /Die-1/. It uses only one parameter \( R \) which is the ratio between the characteristic dimensions of the current design (to be estimated) and the basic design. The function has been derived for a shaft with flange using similarity characteristics:

\[
\text{COST} = 72.8 \times (0.44*R^{2.3} + 0.34*R^{18} + 0.20*R^{0.9} + 0.02)
\]

Other examples of cost functions can be found in /Rie-1/ (general), /Ric-1/ (turned and milled parts), /Mah-1/(turned parts), /Pah-2,Bei-1/ (welding), /Pah-3/ (rolling mills), /Eve-1/ (fixtures), /Ehr-6,Kie-1/ (gears), /Pac-1/ (moulded parts) and /Bau-1,Kre-1/.

Global cost estimation methods which do not directly use cost functions have been reported by Lorenzen (nomograms for gears, see /Rie-1/), Hillebrand /Hil-1/, Knight and Poli /Kni-1,2/ and Freimann /Fre-1/.
Global cost estimation methods will be applied when insufficient product information or insufficient time is available to perform a detailed cost estimate, or when the potential higher accuracy of a detailed estimate is not required. The methods will be used especially during the Materialization phase of design, to get a fast estimate of the absolute costs or to compare alternative design solutions. (Some of the methods may be applicable on conceptual designs). German statistics showed that global cost estimation methods are the main tool that machine designers desire /Vdi-1/.

2.4.5 Detailed Cost Estimation

In this report detailed cost estimation will imply that complex products are estimated by distinguishing all the simple component parts and all necessary (assembly) operations. Costs for component parts are estimated considering the basic materials and the operations necessary to produce it. Material costs are estimated in detail, taking into account the waste produced. Operation costs are estimated using operation times and hourly rates for man and machine. The use of times and rates is typical for detailed estimation.

The above definition implies that detailed cost estimation requires complete product data (geometry, material, quantity to produce) and a detailed process plan as input. Therefore the method is applicable only during (or after) the detailed phase of design. As the method requires detailed process planning, tedious calculations, search for basic data and some experience, it is not useful for the designer unless it can be automated to a considerable extent. Traditionally the method is applied by specialists in the cost estimating departments for pre- and post-production calculation.

Detailed cost estimation will typically be used to calculate absolute costs for a design. The method provides quantitative information. It should be noted however that detailed cost information may be necessary for the derivation of cost structures, relative costs or cost functions.

2.5 Some problems associated with Cost Information

2.5.1 Conceptual Designs

From the point of view of the development of cost information tools some fundamental difficulties are associated with the conceptual phase of design. During this phase the possibilities to influence costs are many and the costs associated with a change in the design are still low. Consequently, cost information is of prime importance here. Due to the fact that very little is known about the product however, it is very difficult to estimate the costs of a conceptual design in a meaningful way. Usually, at least some form of materialization is required to apply existing cost estimation techniques.

Yet, some authors present systems or methods for cost estimation of conceptual designs. Examples include the Search Calculation /Hil-1,Ehr-9/, the work of Ferreirinha on milling machines /Fer-1,2/, the PRICE-program /Fre-1/, the cost data booklet on power transmission systems by Culley /Cul-1/ and other methods reported by Ostwald /Ost-2/. Note however that the term
'conceptual design' is ill-defined and probably has a different meaning for the different authors. Some of the methods will be discussed briefly in the next chapter.
In addition, although design methodology describes how the designer should work, we do not know very well how he or she actually proceeds in this phase /Rut-1, Ull-1/. This makes it difficult to develop adequate tools.

2.5.2 Required organizational measures

Organizational measures create the environment in which cost and manufacturability information is prepared and used. As it is not the main field of interest here, some important points will be stressed only to show that it is a subject that should not be forgotten when discussing cost information for the designer.

Primarily, organizational measures have to ensure the production of cost information. Gathering time and cost data, processing it to obtain suitable cost information for the designer and keeping this information up-to-date requires time and money. This can be conceived as a long term investment. Company management should ensure the availability of adequate resources for these tasks, even under the pressure of many short term problems that have to be dealt with (/Ehr-1, Vdi-1/).

Secondarily, management should ensure the use of the cost information. Every design assignment should have target costs associated with it and costs should be checked several times during the design process (‘mitlaufende Kalkulation’) /Ehr-1/.

Other ‘organizational’ measures are associated with:
- clearing the ‘mental walls’ between departments,
- lifting the ‘curtains of secrecy’ surrounding cost information,
- standardization, classification and use of subassemblies as ‘building blocks’,
- institutionalization of contacts between the design department and the process planning and cost estimating departments,
- simultaneous engineering,
- structuring of the design process.
Additional remarks on the topic can be found in /Ehr-1, Vdi-2, Vdi-3, Rad-1, Rot-1/.

2.5.3 Cost Information is not always economical

Cost information can be used by the designer to save money. On the other hand gathering time and cost data, preparing it for use by the designer and teaching the designers how to use it effectively will cost money. This implies that it is not always economical to supply cost information. Important decisions are necessary to select the types of objects for which cost information will be supplied and to select the type and amount of this information. Two points need attention: the savings obtained every time the information is used and the number of times the information can be used. Usually the two will be
complementary: information that can lead to high potential savings will probably not be applicable often and the other way around.

This topic has been discussed by Radermacher /Rad-1/ and Schuppar /Sch-2/.

2.5.4 Time dependency of the information

Cost information should be kept up-to-date. Probably this is one of the most awkward problems in supplying cost information to the designer. Changes will occur in costs of materials, wages, interest rates, rules for depreciation, technological data (cutting speeds), calculation of machine rates and the like. Cost information should be updated regularly to reflect these changes. This fact should be taken into account when designing the system to generate the information (use of computerized techniques) and when making the costs/savings analysis.

The subject is often discussed in relation to the choice between relative and absolute costs. The latter are known to be very subject to change. Relative costs are often said to be more stable. Claussen /Cla-1/ is very sure of this: 'History shows that in spite of wars and disasters and in spite of political, technical and economical revolutions, relative costs and cost structures defined in the right way will remain almost constant for periods that will usually be considerably longer than the life time of the products'. This is clearly exaggerated. In general the topic is disputed, see /Spu-1,Kre-1/ for example, and many authors prefer absolute costs.

More information on the time dependency of cost information can be found in /Ehr-8/.

2.5.5 Company dependency of information

Cost information will usually be strongly company dependent. Ehrlenspiel /Ehr-8/ shows that estimates made by different companies can differ as much as a factor four. This implies that most cost information will have to be generated by the companies themselves. Work in research institutes like universities will be limited to the development of methods to prepare the information and to keep it up-to-date and to the description of the organizational measures required.

Cost information that depends on the market in general or on common technological factors may have a more general validity. According to Ehrlenspiel this includes check lists for cost effective design, main parameters that influence costs, cost structures, global design rules, similarity rules, methods to prepare and maintain cost information and relative cost catalogues for standard parts and materials /Ehr-8/.
2.5.6 Accuracy of global cost estimation methods

Ehrenspiehl states (Ehr-8) that a difference of 20 to 30% between the results of global cost estimation methods and pre-production (detailed) estimates is acceptable. The general impression from all literature studied is that an accuracy of 10% is usually aimed at and that most methods meet this demand. Ehrenspiehl analysed the difference with results from pre-production estimates of gears for five types of global estimation methods (Ehr-6).

He found for instance:
- for the weight method an average difference of 9.3%,
- for cost functions based on similarity characteristics an average difference of 4.7%,
- for functions based on regression analysis an average deviation of 2.6%.

This demonstrates that global cost estimates can be quite accurate when the basic data are prepared with care and used in the proper way.

Additional information on this topic can be found in (Ehr-6, Ehr-8, Ver-1, Bur-1, Fis-1).

2.6 Summary

This report is restricted to financial costs. Psychological, sociological and environmental product costs are not discussed. Emphasis is on manufacturing costs. Little or no attention will be given to costs for marketing, distribution, the design activity and other company costs.

The Basic Design Cycle illustrates that the designer has to use cost information in an iterative process aimed at cost effective design. This implies that the information has to be related to factors that can be influenced by the designer. Apart from telling how much a (partial) design will cost, the information should also indicate how the design is to be adapted to make it cheaper.

The Three Phase Model of the design process shows that the designer needs different types of cost information for different partial design tasks.

Cost structures reveal the factors, types of costs or parts of the design that will have a major influence on the eventual product costs. This enables the designer to devote most of his time to the most important aspects. Relative (or proportional) costs allow the designer to select rapidly the cheapest alternative. Design rules, to be conceived as manufacturability information, aid the designer in avoiding expensive solutions. These types of information help in obtaining cost effective designs but they do not quantify the savings. Hence the introduction of the concept 'qualitative cost information'.

Global or detailed cost estimation methods can tell the designer how much a design will cost or how much can be saved by specified changes in the design. This 'quantitative cost information' allows the designer to check whether target costs will be met or to compare alternative designs.

Cost information can only be effective if organizational measures ensure its generation, maintenance and use.
3. Tools and techniques

The preceding chapter describes the types of cost information that a designer will need. In this chapter we will consider some techniques and tools for the preparation and use of this information. The discussion is restricted to computerized tools with an emphasis on those linked to CAD-systems. Additional general information on cost information tools for the designer can be found in /Kie-1, Rau-1, Ehr-2, Wie-3,4/.

3.1 Qualitative cost information tools

As indicated before, qualitative cost information guides the designer towards a cost effective design without quantifying the savings. The most important types of information in this group are manufacturability information, cost structures and relative costs. It will be clear from the preceding chapter however that a wide variety of information can be imagined here.

3.1.1 The passive approach

Designers now obtain qualitative cost information by consulting catalogues, handbooks, colleagues, process planners, cost estimators and the like. In a computerized tool these sources will have to be replaced by a database (figure 3.1).

Apart from the decision which information is to be stored (economical considerations) and the gathering and preparation of this information, the creation of such a tool is mainly a database design and management problem. Considering the amount of data, the wide variety in types of data, the complex relations between the data and the time and company dependency of the information, the design and maintenance of this database is not an easy task. In general the development of large databases for engineering and design applications still presents problems /Reh-1, Ben-1, Mcc-1, Vro-1, Sta-1/ and research is ongoing to solve these. Moreover a lot of attention will have to be paid to the design of the user interface, both to facilitate the specification of a request for data and to obtain a clear presentation of the data. This topic will not be discussed any further.

![Diagram](image)

**Figure 3.1:** Database search for qualitative cost information. The lack of suitable databases for engineering and design hampers the development of large integrated data collections. A good user interface is of prime importance.
Figure 3.2: Programs that check designs for manufacturability are usually based on features. These are partial forms or other product characteristics that are considered as a unit during design, process planning or cost estimating. Recognition of features in the CAD description for a part is one of the problems involved.

3.1.2 Some active approaches

Passive tools, although potentially very useful, do not permit anything else than a database search for information. The designer has to take the initiative to use the tool and he has to specify what he is looking for.

A more active approach is possible by the development of programs that control the manufacturability of a design, either continuously during the design process or at moments selected by the designer. In other words, these programs try to check designs for violation of design rules.

Usually the concept 'feature' plays a central role in such programs. Although there is no consensus on the definition, a feature can be conceived as any partial form or product characteristic that is considered as a unit during design, process planning or cost estimating. Form features describe part of the geometry of a design (holes, slots, faces), but non-geometrical features (tolerances, surface qualities, heat treatments, material) can be defined as well.

Two approaches to the control for manufacturability are possible. In the first approach the designer is allowed to use only those features that can be manufactured. Other features are not offered in the CAD-system. Delbressine /Del-1/ used this approach. The second approach allows the designer a 'complete modelling freedom'. A feature recognition program is used to find the features in the design (figure 3.2). Currently, both approaches present problems. Feature recognition is still a bottleneck. In addition Delbressine found that it is not so easy to define when features can (easily) be manufactured. See paragraph 4.5 for some additional remarks on the topic.

Nevertheless noteworthy attempts to develop 'design for manufacturability' programs have been made by Delbressine /Del-1/, Swift /Swi-1/, Schnitz /Sch-9/ and Lim /Lim-1/. In addition the literature on Computer Aided Process Planning (CAPP), a closely related topic, provides a good impression of the problems and possibilities (see /Erv-1/ for example).

If a program would actually understand what a designer is doing at a certain moment, it could provide a more active assistance. It could volunteer information, indicate the next step in the design process, comment on the design or point out alternatives even without being asked to do so. At the
moment this idea is quite futuristic, but research on Design Methodology and Artificial Intelligence may eventually provide some solutions. Although not always directly related to cost effective design, the following references give an impression of the possibilities and problems: /Sto-1, Lim-1, Kal-1, Ull-1, Fra-2, Bro-1, Wit-1, Wie-6, Swi-1, Jak-1/.  

3.2 Global Quantitative Cost Information

As mentioned in paragraph 2.4.4, global cost estimation methods are one of the main tools that machine designers desire /Vdi-1/. Consequently a lot of research has been done to provide these methods. Cost Functions or Fast Calculation Formula's (german: Kurzkalkulationformeln) are probably the most important examples.

3.2.1 Cost Functions

In general a Cost Function is an arithmetic expression that contains parameters and constants (see examples in 2.4.4). The function is valid for a class of objects. The parameters in the function represent the factors that influence the costs. Hence the types of parameters depend on the class. Usually the number of parameters will be small to make the function easy to apply and to show the influence of the most important factors in a clear way. The constants in the function quantify the influences of the parameters on the costs. Their values will depend on the class.

Cost Increase Functions (german: Kostenwachstumsgesetze) are a special type of cost function. They are valid for restricted classes of objects that are very similar both in geometry and in manufacturing processes used. Typically they contain one function parameter, the characteristic object property, only. The function shows how costs 'increase' depending on the value of this characteristic parameter.

The Unit- or Factor methods mentioned by Ostwald /Ost-2/ can be conceived as very simple cost functions. Probably the most well known example is the Weight method which uses the weight (mass) as the function parameter. This parameter is multiplied by the costs per unit of weight (kilo) for the appropriate class of objects. The method has to be applied with care /Ehr-6/ but can be reasonably accurate (2.5.6).

The method of Material Cost Shares can also be conceived as a cost function. It uses the material costs as the only parameter and the ratio between material costs and total product costs (the material cost share) as the class dependent constant. The function divides the parameter by the constant. The method is recommended by the german association of engineers for complex products (see details and some data on material cost shares in /Vdi-4/). The 1-3-9-Rule presented by Rondeau is related to this method /Ron-1/.
3.2.2 Derivation and use of Cost Functions

Basic research on the derivation of cost functions has been done in Germany. Taking the description provided by Eversheim /Eve-1/ and Fischer /Fis-1/ as a guide, the following steps can be distinguished (figure 3.3):

- consider a large number of existing products (or other objects) and define classes. One of the techniques used to find suitable classes is Cluster Analysis /Fre-2, Kre-1/. Good class definitions are essential to obtain a function with high accuracy;
- choose potential factors that influence costs. Preferably the factors should be object properties that can be quantified (dimensions, tolerances, number of components), and that can be influenced by the designer in a direct way;
- examine whether the chosen factors really influence costs by applying regression analysis or parameter optimization techniques. This reveals the types of parameters and their influence (the values for the constants in the cost function) /Eve-1, Die-1, Ehr-7, Bau-1, Kre-1/;
- derive the cost function using only the parameters with the largest influence. (The number of parameters determines the applicability and the accuracy of the function);
- test the function by applying it on the existing products. The differences between function results and known real costs will reveal the function accuracy.

\[
\text{Stored data on existing products (geometrical data, manufacturing data, costs)}
\]

\[\text{Select Objects for which Cost Info is economical}\]

\[\text{Selected Objects (products, components, features)}\]

\[\text{Classification using Cluster Analysis}\]

\[\text{Classes of selected Objects}\]

\[\text{Determine types of function parameters (P) and values of function constants (C,E) using regression analysis}\]

\[\text{Cost Function per Class, COST} = \sum_{i} (C_{i} \cdot p_{i})\]

\[\text{Test function for existing objects}\]

Figure 3.3: Main steps during the derivation of Cost Functions. Such functions supply global quantitative cost information and are easy to use by designers.
<table>
<thead>
<tr>
<th>Operation</th>
<th>Costs increasing with $R^3$</th>
<th>Costs increasing with $R^2$</th>
<th>Costs increasing with $R$</th>
<th>Constant costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Material</td>
<td>800</td>
<td>500</td>
<td>60</td>
<td>15</td>
</tr>
<tr>
<td>Burning</td>
<td></td>
<td></td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>Bevelling</td>
<td></td>
<td></td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>Affixing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Welding</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Annealing</td>
<td>80</td>
<td></td>
<td>40</td>
<td>70</td>
</tr>
<tr>
<td>Sandblasting</td>
<td>40</td>
<td></td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Marking</td>
<td></td>
<td></td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Drilling</td>
<td></td>
<td></td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>De-burring</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_b = 1890$</td>
<td>$\Sigma_3 = 920$</td>
<td>$\Sigma_2 = 500$</td>
<td>$\Sigma_1 = 370$</td>
<td>$\Sigma_0 = 100$</td>
</tr>
<tr>
<td>$\Sigma_3 / C_b = 0.49$</td>
<td>$\Sigma_2 / C_b = 0.26$</td>
<td>$\Sigma_1 / C_b = 0.20$</td>
<td>$\Sigma_0 / C_b = 0.05$</td>
<td></td>
</tr>
</tbody>
</table>

With the resulting cost function: $C_n = (0.49R^3 + 0.26R^2 + 0.20R + 0.05)C_b$

**Figure 3.4:** Derivation of a cost increase function based on similarity characteristics as proposed by Pahl and Rieg. $C_b$ are the costs for the basic design, $C_n$ the costs for the new design and $R$ is the ratio of the characteristic parameters for new and basic design. (Source: /Pah-1/, adapted)

Diels uses an approach similar to the one described above for the derivation of cost increase functions, where values for the constants are determined using regression analysis on data for a large number of existing products /Die-1/. Pahl and Rieg /Rie-1,Pah-1/ start from only one known design. They apply similarity characteristics on manufacturing times to analyse what parts of the costs vary with the third, the second or the first power of the characteristic parameter and what parts are constant. This is illustrated in figure 3.4.

An alternative way to derive cost functions is the 'Bemessungslehre'. The method was originally proposed by Kesselring /Kes-1,2/. It tries to combine technical and economical factors into one equation. Clearly this is useful for the designer, but the function tends to be rather complex /Rie-1,Han-1/. The method is recommended by the german association of engineers for less complex products, as an alternative for the method of material cost shares. Details and examples can be found in /Vdi-4/.

The use of cost functions is illustrated in figure 3.5 and includes the following steps:
- determine to which existing class the current design belongs and get the associated cost function,
- determine the values of the parameters in the cost function,
- evaluate the function result.
Important for the derivation of cost functions is the structured storage of cost data on existing designs and the statistical processing of these data using regression analysis, parameter optimization techniques and cluster analysis. Computer programs including these tasks are:

- the RELKT program by Beitz and Klasmeier /Bei-1/ which includes structured storage of cost data, statistical processing and presentation of cost functions and relative costs,
- the RELKO program developed by Spur and Kreisfeld /Spu-1/ for regression analysis on existing cost data, the determination of parameters that influence costs and the presentation of cost structures,
- the KOINF-program by Kreisfeld /Kre-1/.

### 3.2.3 The KOINF program by Kreisfeld

KOINF /Kre-1/ consists of a part to prepare basic cost information data and another part to use these data to get a cost estimate. Both parts are linked to the COMPAC CAD-system and restricted to turned parts. Some exceptions to the rotational symmetry are allowed however.

Part geometry is described using a main shape and additional detailed shapes (elements or features). The main shape is described by axial and radial surfaces for which axial length, inner- and outer diameters and location (inside, outside) are used as parameters. The detailed shapes have separate, shape dependent, descriptions. Tolerances and surface qualities are taken into account by deriving an operation requirements value. The influence of type of material is represented by the definition of material classes.
The assumption is that geometries of existing parts are stored in the CAD-system and that the associated process planning and cost data are also available. The data preparation part of KOINF will first derive the KOINF-geometry description from the CAD-representation for the part. Next the material class is determined, followed by the operation requirements values per element. Cost data are also stored per element. Using multiple regression techniques and cluster analysis the main cost influencing factors are derived and cost functions for classes of shape elements determined.

Cost information derived and stored in this way can be used in two ways. A global cost estimate can be obtained by manual input of the values of the main cost influencing parameters. A more detailed estimate (although still using a cost function) is based on all influencing parameters. Geometry for a part to be estimated can be taken from the CAD-system. The part description is interpreted, the class to which the part belongs is determined and an estimate is performed using the cost information for the class. Output shows the costs for the main element and for the additional detailed shape elements separately. Moreover the KOINF program can provide recommendations by showing the effect of changes in design parameters on part costs. Changes are not automatically reflected in the CAD-description of the part however.

3.2.4 Other programs for global cost information

In his Search Calculation, Hillebrand defines a multi-dimensional space of characteristics in which objects can be placed using a location vector /Hil-1/. A distance between any two objects in the space is defined. This distance serves as a measure for the extent of similarity of objects. The object to be estimated is placed in the space according to its characteristics and the distances to other objects are determined. The costs of the object are now estimated to be equal to those of the nearest (that is the most similar) object. According to Ehrnienspiel /Ehr-9/ this ‘Suchkalkulation’ can also be used for conceptual designs.

Knight and Poli developed a program to estimate forging costs /Kni-1,2/. A one digit code for materials and a three digit code for part shape and size, both indicating forging difficulty in some way, provide access to a database with relative forging costs. Information in this database has been derived by analysis of existing forged products. Costs of a design are obtained split in material-, operation- and die-costs, and in proportion to the costs for a specified basic design. Using company specific costs for this basic design and an assumption on the average shares of the three types of costs in the total costs, an absolute cost estimate can be obtained. The program is part of a larger Design for Forging System. The approach is said to be applicable on other manufacturing processes as well.

One of the major programs on the other side of the Atlantic is PRICE /Fre-1/. It is a very extensive program that uses historical data to estimate system costs during conceptual design. Although intended for use in the design department, its level of abstraction is quite different from the examples mentioned above. PRICE uses the parametric cost model (similar to the cost function approach,
see also /Apg-1,Mal-2/) and expects input like product size, weight, type of componentry, power dissipation and prototype and production quantities. Clearly it is used to estimate system costs rather than product costs in the sense used in this report. PRICE is used mainly by the US Air force, Army and Navy, NASA and similar institutes.

Figure 3.6: Main steps in the derivation of basic information for detailed cost estimation. Although tedious and time-consuming, techniques to derive the information are available. Usually one of the main problems is the determination of operator dependent times.
3.3 Detailed Quantitative Cost Information

A global cost estimation method is always defined for a class of objects. Although operation sequences (process plans) may be used in the derivation of the methods, they are usually not necessary during the use of the methods. Figure 3.5 illustrates this. In detailed cost estimation however, the operations necessary to produce an object play a central role.

3.3.1 Derivation and use of detailed cost information

The main steps during the preparation of the basic information for detailed cost estimation are shown in figure 3.6. The main problem is probably to find reliable normal- or standard times and to define time allowance factors. Machine dependent times can usually be calculated. Operator dependent times can be measured in practice using time study analysis, they may be determined using methods like Work Factor or Methods Time Measurement /Mal-t/Wil-1/; they
can be found in literature or the information can be bought /Ydo-1, Ost-1/. Examples for the determination of rates (costs per unit of time) for operator and machine can be found in /Vdl-5, Koo-1/ and other literature on business economics. One of the main problems may be to decide which costs are to be included in the rates.

Clearly, the preparation of all this basic information is a tedious and time-consuming task, but in principle all tools and techniques required are available.

The detailed estimation itself includes the main steps shown in figure 3.7. The main problem is the process planning, especially the detailed process planning. The designer will probably have some idea of the required operations and their sequence (the global process plan). Asking him to determine all partial actions within the operations and the value of all parameters needed in time- and tool estimating, will usually be asking too much. This detailed process planning has to be automated to a considerable extent before detailed cost estimation methods can be used by the designer.

3.3.2 Ferreirinha: the HKB—program

HKB is a program for detailed cost estimation developed by MIRAKON in Switzerland /Fer-3, Mir-1/. The program is based on the theory developed at the ETH of Zürich by Ott and Hubka /Ott-1/. Although it performs detailed cost estimation it can be used by designers. The process planning problem has been solved by predefining partial process plans per feature.

The program offers a framework in which a company has to insert the following basic data /Fle-1/:

- a list of form features containing all features used in the company. Per feature parameters describing geometry, tolerances and surface qualities and operations required to produce it have to be defined. The latter can be made to depend on the parameters.
- a list of all operations used in the company. Associated information includes set-up time, main- and secondary time, time-independent costs and tools to be used,
- a list of all types of manufacturing processes in the company with the related machines or working places,
- a list of groups of machines and/or working places. Associated with every machine is information on special suitability, technical limitations, costs per hour and the like,
- a list of all materials with their corresponding properties and special information related to the use of the materials in combination with specific tools or manufacturing procedures.

The logical combination between these tables is obtained by entering the same parameters in different lists, either as a criterion for decision or in some formula.

The designer does not see the above lists and relations. He or she enters product geometry and additional data in dialogue with the program by choosing from menu's on the screen. Form features are also selected in this way and are represented graphically on the screen after selection to facilitate additional...
input. Alternatively, the HKB-program can accept the part geometry from the CADAM CAD-system, through a special interface.
Output of the program includes operation times per formfeature divided in setup, main, secondary and total time, and costs for the part. Iterative optimization is possible.
According to Ehranspiel /Ehr-9/ the HKB-program will typically be used during process planning, although it can be useful during design.

3.3.3 Programs by Baumann

Baumann /Bau-1/ developed a set of programs integrated with CAD that provide cost information on parts made by metal removal operations like turning, milling, drilling and grinding (the programs seem to have been implemented for turned parts only). Bas for the programs is a detailed cost model that uses a process plan and operation times. The partial operations (actions) to be distinguished and the associated times have been determined.

![Diagram of a mechanical part with notes on cost savings and machining times.]

**Figure 3.8:** Output of Baumann's cost estimation program linked to CAD. Cost information is associated directly with formfeatures that can be chosen from a menu in a 2D CAD-system. The program can generate design variants automatically and indicate the savings they will bring. If the designer accepts the change, the CAD-representation is updated automatically. (Source:/Bau-1/,adapted)
by statistical analysis of the data on a large number of existing products. Direct use of this detailed cost model requires considerable manufacturing and process planning knowledge. The designer has to indicate what operations will be used and which form features are to be produced by them. This is clearly a weak point.

Strong features of Baumann's program include its integration with CAD and its ability to generate recommendations for redesign. The CAD-system used is 2D and operates with a menu from which basic shapes like cylinders or cones can be selected. Thus a dimensioned turned part consisting of form features can be defined. After definition of the operations and the input of additional data like material type and quantity of parts, a detailed cost estimate can be obtained. Next the program will search for variant designs that lead to savings and present the possibilities to the designer (figure 3.8). If the designer accepts the change, the CAD representation is adapted automatically. Within limitations the program can also be used when only the part shape is available and not the dimensions.

In addition Baumann derived global estimation functions (for manufacturing times) and general design rules from the detailed cost model. The time functions require a general input consisting of length, width and height of the part, quantity to be produced and the number of form features for every type of operation (counted in a special way). He discusses the formulation, classification, retrieval and use of global design rules.

3.3.4 Other programs for detailed cost estimation

A large quantity of software for or related to detailed product cost estimation is available. Most of these tools are not suitable for use by the designer however. They have been developed for use by process planners or cost estimators and require the corresponding detailed manufacturing and estimating knowledge. The cost information provided is usually not related to parameters that can be influenced by the designer and therefore not suitable for use in the design department in the way that it should be. Examples of this type of software include the BASIC programs described by Nicks /Nic-1/, the packages reviewed by the American Machnivist /Ame-1/ such as the AM Cost Estimator /Ost-1/, the Costimulator from Manufacture Technologies described by Casey /Cas-1/, the YDO-Calc programs /Ydo-1/, Melzer's program to estimate costs of injection moulded parts /Mel-1/ and the software currently being developed in Lotus 1-2-3 by the Massachusetts Institute of Technology.

Although it is linked to CAD, the Turned Parts Estimating package described by Astrop /Ast-1/ also belongs to this category. The user has to 'machine' the component step-by-step on the screen to generate the manufacturing data used in the estimate. Clearly this requires process planning knowledge not available to the designer.

Schouenberg's programs for the design and economical manufacture of injection moulded parts should probably be used by designers and process planners together /Sch-3..8/.
Ehrlenspiel /Ehr-9/ describes the GUSSKAL program developed by Pickel. The program calculates costs for cast steel parts which includes die construction, the casting operation and metal removal operations on the casting. A turbine blade is presented as a typical example. Geometrical data needed for the cost estimate are derived automatically from a 3D representation of the object in the CAD-system CV Medusa.

Ehrlenspiel points out that unlike the programs by Baumann and Kreisfeld, GUSSKAL allows complete modelling freedom for the designer.

3.4 The KIS approach by Ehrlenspiel and Hillebrand

The Cost Information System KIS /Ehr-9/ aims to support the designer in all phases of the design process, as described in chapter 2. KIS offers a frame to which all kinds of dedicated cost information- and calculation programs can be linked. It is intended to be linked to CAD-systems and has three main functions: data search, design assessment and cost estimation (figure 3.9).

Currently KIS is linked to the CAD-system Euclid, in which a geometric model can be build in the usual way. Special functions allow the identification of relevant formfeatures and the specification of tolerances and surface qualities. The internal computer representation of the Euclid models has been extended to include these manufacturing data (product model). Closely linked to this Euclid product model is the attribute model in KIS: data can be exchanged in both directions using CAD-system specific link programs. The attribute model consists of production elements and production units. Typically the elements are formfeatures that are relevant for process planning, with their associated

![Diagram](image)

**Figure 3.9:** Main functions in the CAD-linked KIS-program developed by Ehrlenspiel and Hillebrand. KIS is a Cost Information System that aims to support the designer during all phases of the design process. It offers a framework to which dedicated programs can be linked. Main functions include problem oriented search of cost data, cost estimation using either global or detailed methods and design assessment based on a list of demands. (Source:/Ehr-9/, adapted)
data as the attributes. Operations can also be modelled in this way however. Production units represent component parts and their manufacturing. Basically it is also possible to model complete complex products in this way with all their component parts and subassemblies.

The 'data search system' permits a problem oriented retrieval of cost information such as design rules, relative cost tables, lists of standard parts and lists of existing designs. In addition the system supports the selection of manufacturing data like tolerances or surface qualities and the selection of types of operations. When the lists of existing (standard) parts would be extended to include costs, the search system could also be used for Hillebrand's Search Calculation (paragraph 3.2.4) during early phases of the design process.

The 'estimation system' will use the attribute model as input and select a suitable estimation program for each production element in this model. All kinds of calculation and estimation programs can be linked to KIS for this purpose, using either global or detailed methods. Estimation results can be presented per production unit using different selection criteria.

The 'design assessment part' of KIS allows input of a weighted list of demands for the design. Using this list KIS can assess design variants automatically by applying design assessment methods as described in /Vdi-4/. Both technical and economical data are taken into account. Currently KIS is being prepared for commercial sale in cooperation with IBM Germany /Ehr-10/.

3.5 Summary

Qualitative cost information can be supplied in a passive and an active way. The passive approach does not allow anything else than database search for information. More active design assistants may be possible in future, depending on the result of research on Design Methodology and Artificial Intelligence. Some programs that check designs for manufacturability have been written but they seem to be prototypes.

Especially in West-Germany a lot of research has been done to provide global cost estimation tools. Techniques and tools to derive and use cost functions are available. The main advantage of global cost estimation methods as compared to detailed cost estimation is that they are easier to use by designers.

Apart from the tedious and time-consuming preparation of the information, the main problem in detailed cost estimating is the detailed process planning. This has to be automated to a considerable extent before detailed estimation can be applied by the designer.

The 'feature' concept plays a central role in all advanced cost information tools. The lack of suitable databases for design and engineering probably hampers the development of large advanced cost information tools.

Most research on cost effective design has been focused on machine design, on metal removal operations like turning and milling and on production of small quantities of products. This was one of the reasons to concentrate our research on mass production of sheet metal parts.
4. Outlines for the DIDACOE programs

4.1 Introduction

DIDACOE is an abbreviation for Delft Integrated Design And Cost Optimization Environment. It is a research project that started in autumn 1985 at the Faculty for Industrial Design Engineering of the Technical University of Delft. This Faculty focuses on the design of mass produced consumer goods. Its department for Engineering Design has special knowledge on the design and production of parts made from sheet metal or from engineering plastics. Cost estimating activities are closely linked to the sheet metal specialists. Consequently some detailed cost estimating programs for stamping, shearing and deepdrawing had already been developed before the start of the DIDACOE-project /Her-1,Haa-1,2,Boo-1/. The original objective of this project was to extend these programs and to link them to Computer Visions CAD-system Medusa. The idea to explore the possibilities for a completely integrated design and cost optimization environment was added later. This short background description explains the major features of the DIDACOE-project: mass production, detailed cost estimation, sheet metal parts and CAD. These characteristics, their consequences and some associated topics will be discussed in relation to the contents of the previous chapters.

4.2 Mass production

Most research on cost information for the designer as described in the previous chapters focused on small quantity production, machine design and metal removal operations like turning, milling and drilling. Comparing these situations to mass production a number of essential differences can be noted.

In mass production, more time and money can be spent to optimize the designs. Because of the large quantities such investments will pay off even if the savings obtained per piece are small. This implies that there is more time to complete the long cost control circuit as described in paragraph 1.3. Indirectly, due to the more frequent contacts between design and process planning, this will probably also reduce the communication problems. Thus there is less need to develop global cost estimation methods for the designer. In addition the potential higher accuracy of detailed cost estimation is required to recognize even small savings.

Maybe it is also more difficult to develop cost functions in mass production situations. The lack of time in small quantity production more or less implies a need to reuse existing partial design- and manufacturing solutions. Consequently it is comparatively easy to construct limited classes with a number of existing objects that is sufficient to build reasonably accurate cost functions. Mass production is more characterized by the need for optimization of each new design. Potentially the variety in design- and manufacturing solutions is larger and it may be more difficult to define the required classes indicated above.
Another main difference is the extent of integration of the manufacturing processes. Small quantity production is usually characterized by a sequence of operations in which standard tools are used. Typically, different form features will be manufactured sequentially with an almost one-to-one relation between form features and operations. Mass production shows a more integrated manufacturing, where many form features are produced simultaneously using special tools. This implies large differences in process planning problems and in the possibilities to relate quantitative cost information to form features.

The use of special tools in mass production implies that tool costs are a major factor to consider. Most existing cost information programs hardly consider this aspect: the costs of standard tools used in small quantity production are usually 'hidden' in the machine rate or they are an integral part of the operation costs.

In addition, material costs received little attention in most of the existing programs. Both Baumann /Bau-1/ and Kreisfeld /Kre-1/ ignore them completely. Baumann motivates this by pointing out that the share of material costs in the total costs was less than five percent for 90% of the products that he considered. Kreisfeld does not include these cost because they can easily be calculated using existing methods. Thus, in most of the existing programs, design-oriented cost information is mainly based on operation costs. This is especially true for the feature-related quantitative cost information. In mass production situations where material costs, operation costs and tool costs have to be optimized together such limitations are usually not acceptable.

### 4.3 Cost estimation

The main objective of cost estimation is the determination of the value of the attribute 'cost' for some object. Simplifying the real possibilities, four main object levels (or estimation levels) can be distinguished: the product level, the component level, the operation level and the action level (figure 4.1). On any level, objects can be estimated globally or in detail. Global estimation is mainly characterized by a direct estimation of an object on its level. Typically product costs could be estimated by a cost function that contains only product characteristics such as mass, length and height as parameters. Detailed estimation is mainly characterized by indirect estimation of an object on levels that are lower than that of the object itself. This implies that an object analysis is required to find its components and operations. Product costs are now estimated as the sum of the costs of these elements.

Considering available experience and the requirements of mass production it was decided to take detailed cost estimation as the base for DIDACOE. In principle this implies that estimation is performed on the action level. In the DIDACOE cost estimation concept, global estimation is allowed on this level only. Tool costs could be estimated using a cost function for example. Some attention was paid however to the extension of the concept to global cost estimation on higher levels.

It is possible to define a general detailed cost estimation concept valid for (almost) all products, components and operations. This concept will be described in the next chapter. On the action level however the solution of
Figure 4.1: Four main levels of objects for cost estimation. DIDACOE estimates costs of simple components in detail by distinguishing their elements on the action level. Although desirable, it is difficult to assign complete quantitative cost information to functional units or to features.

estimation problems depends heavily on the type of operation. Specific procedures have to be developed per operation for the estimation of times and tool costs. An example for the production of sheet metal parts from strip using progressive dies is presented in chapter 6.

Although the detailed estimation concept also deals with product estimation, the DIDACOE-project is essentially limited to optimization of costs for components ("Einzelteile"). Basically this presents problems. Several components will usually interact within a larger complex product. Hence minimizing the costs for one component may cause an increase in costs for adjoining parts. As noted by Radermacher /Rad-1,2/ and Ferreirinha /Fer-1,2/ the use of so called functional units (collections of parts and features that together perform one function) could overcome this problem. It is difficult however to define the units properly, to assign costs to them and to recognize them in CAD-models.

4.4 Process planning

A detailed cost estimation program needs a completely detailed process plan as input. This plan can be imagined to consist of two parts.
The global process plan specifies the material from which production starts (at least the type of shape) and the required operations and their sequence. Operations should be defined in such a way that they imply at least a type of machine and a type of tool. An example could be:
(1) shear strips from sheet using a power shear,
(2) stamp products from strip using a progressive die, and
(3) finish surface by electroplating.
Detailed process plans per operation describe all the operation details: speeds and feeds, press strokes per minute, specific machines, tools and fixtures to use, strip layout, etc.
A designer will know the manufacturing principle for his design. That is: he will be able to indicate whether the main character of the production will be metal removal, injection moulding, die casting, stamping or the like. Without such knowledge it is impossible to make a reasonable design. It is already doubtful if a designer can provide a good global process plan. That will depend to a considerable extent on his experience. Usually however, even an experienced designer can not be expected to provide detailed process planning data. Hence, for a detailed cost estimation tool to be useful for the designer, at least the detailed process planning should be automated.

In addition it should be noted that cost information tools are intended to guide the designer, to comment on his design, to teach him something. From this point of view it is wrong to start with the assumption that the designer already knows a lot. Thus, the choice for detailed cost estimation implies that DIDACOE has to include both global and detailed process planning modules. The latter have to be developed per operation, the former per manufacturing principle.

Two main types of computer aided (automated) process planning (CAPP) systems can be distinguished /Sub-1;Erv-2;Har-1/: variant-type systems and generative systems.

Variant-type systems are based on the storage of existing 'template' process plans for groups of similar products. After classification of a new product the corresponding process plan for the class can be retrieved and slightly adapted if necessary. Although variant-type process planning systems are probably the most useful type of system at the moment, they are not expected to meet future needs. The final modification of the template process plan requires human interaction, which implies that variant-type systems can not be used in a highly automated environment and that they are less useful for designers (see paragraph 10.2.1 as well).

The generative approach to process planning focuses on capturing the decision-making logic used by experienced process planners rather than simply recording previously generated plans. It is a system that automatically synthesizes product and process information to create a suitable process plan for a new part /Sub-1/. This generative approach is more suitable for designers and it offers more opportunities to include automatic generation of manufacturability information. Due to lack of a sufficient number of comparable designs and due to the higher need for optimization, it may be the only feasible approach in mass production.

Consequently, environments like DIDACOE seem to need generative process planning systems. The development of such systems is a widespread international research topic. In most cases Artificial Intelligence (AI) techniques are used in an effort to solve the problems. It was decided to do this in DIDACOE as well. This choice is explained in paragraph 9.2.
4.5 The use of features

Although the feature-concept is widely used, it is not well defined. For the time being it is sufficient to conceive a feature as any part of the component geometry or any component characteristic that is considered as a unit during design, process planning or cost estimating. Usually features are partial forms like holes, slots, pockets and notches (figure 4.2), but they can be used for non-geometrical information like tolerances, surface qualities, heat treatments, and material as well.

Features play an important role in human reasoning processes and in computer programs that try to do part of this reasoning. Designers will probably think of a design in terms of function-oriented features while process planners reason about manufacturing-oriented features. Depending on the 'level' of reasoning, features are considered globally or in detail. Therefore it is common to distinguish several levels of features, ordered in a hierarchy or taxonomy (figure 4.3).

Figure 4.2: Example of a component and its feature model. Features usually describe a part of a component geometry. They can be function-oriented (for the designer) or manufacturing-oriented (for the process planner). Non-geometrical features can also be defined. (Source: /Man-2/)
Figure 4.3: Actual features (at the bottom of the tree) are instances of classes which describe types of features. Classes can be ordered in a hierarchy. More general classes are in the top of the tree, specific ones near the bottom. (Source: /Man-1/)

Next to this function in the reasoning process, features are associated with a type of datastructure. They are the basic units for which data (and procedures) are stored.

The choice for generative process planning and the desire to generate manufacturability information imply a need for a part description by means of high-level features (features that have an engineering meaning). Geometric modelling systems usually describe a component using low-level features like points, lines and faces, or blocks and cylinders. A feature recognition procedure is required to extract the high-level features from these low-level CAD descriptions.

As shown by Van 't Erve /Erv-2/ and Delbressine /Del-1/, 2D-representations and 3D wire-frame representations can be ambiguous and contain errors. According to them these representations do not provide a suitable base for
feature recognition. Three dimensional solid models using the Boundary Representation (BREP) or the Constructive Solid Geometry Representation (CSG) /Bae-1, Req-1/ are better suited, as they guarantee geometrical consistency.

As shown by Joshi /Jos-1/, Choi /Cho-1/, Karra /Kar-1/ and Van 't Erve /Erv-1,2/ it is possible to recognize features in a BREP-model. The recognition process is in fact a search for faces that together form predefined features (figure 4.4). It is still difficult however to recognize more complex features. The complexity of the required algorithms grows exponentially with the complexity of the features that have to be recognized /Erv-2/.

Potentially, the CSG-representation seems to offer better opportunities. The so called primitives (blocks, cylinders, cones and the like) come close to the features to recognize, closer anyway than the BREP-faces. In addition, the modelling operations used by the designer, which are stored in the CSG-datastructure, resemble the required manufacturing operations. As shown by Lee /Lee-1/ and Woo /Woo-1/ however, feature recognition in a CSG-model also presents problems. Complex features can usually be constructed in several ways, using different primitives and different boolean operations. A feature recognition program has to recognize all these different representations for the same feature, and that presents a major difficulty.

![Diagram showing various primitives and their sections](image)

**Figure 4.4:** Recognition of the feature 'hole' in a Boundary Representation model. It is a search for faces that together form the feature. First a Hole-Starting-Surface (HSS) has to be found, then Hole-Element-Surfaces (HES) and then a Hole-Bottom-Surface (HBS). Feature recognition is a bottleneck for feature-based process planning. (Source: /Cho-1/)
Consequently feature recognition is considered as a serious bottleneck for automation of the process planning /Man-2/. Of course, manual identification of the features can be used as an alternative. This represents an additional task for the designer however (requiring knowledge and time) and it introduces the chance of errors. As a temporary solution it may be useful, but for the future automated concepts it is not the best approach.

A more attractive alternative for feature recognition is feature modelling /Emm-1,Man-1,2/. In this approach the designer creates the CAD-model using high-level design features that have a clear engineering meaning. This can be convenient for designers, as the features match the level of abstraction on which they work. Moreover, the presence of high-level features in the CAD-representation for a part avoids the need to recognize them. The basic idea is to create a product model described in a datastructure based on features, that can contain all design, process planning and cost information.

Apart from the fact that design-by-features involves 'a revolutionary change, possibly requiring decades of research and the education of a new generation of designers' /Erv-2/ this approach does not solve all problems (yet). One of the main questions to answer is probably if there is a difference between design, process planning and cost estimation features. It is not possible to discuss this topic here in detail, but we will give some considerations:

- features required during process planning for a part will depend on manufacturing technology, that will change in time and differ from company to company. Describing a design using such features would not be very flexible. Mäntylä /Man-1/ notes that design features should ideally be related to functional part characteristics only. In particular they should be independent of manufacturing technology. (Considering that a designer should use available manufacturing technology as a constraint this statement raises some questions);

- sets of features developed from a process planning point of view are not always suitable for use by the designer /Har-1/. In Mäntylä's words: 'Unfortunately, forcing designers to describe objects by means of manufacturing features would lead to unnatural design notation' /Man-2/.

- if a designer defines an object using process planning features he will fix the process plan to a considerable extent. This is not always desirable /Man-1,Har-1/. In addition, different feature descriptions for the same object are usually possible. Considering only the set of features defined by the designer unnecessarily constraints process planning. An interesting solution to this problem using feature relaxation is proposed by Mäntylä /Man-2/.

It should be noted that most current work on feature-based process planning and on feature modelling is associated with metal removal operations. In other domains, feature modelling may present new problems.

From the point of view of the development of cost information tools for the designer, features present another problem. To provide the required link between cost information and factors that can be influenced by the designer it would be ideal if this information could be associated with design features. For
qualitative cost information like design rules or manufacturability comments this will usually be possible. Global quantitative cost information is usually made designer-oriented by showing the influences of the parameters in the cost functions. These parameters can be conceived as cost estimation features, which are quite different from the design or process planning features discussed above. Some existing programs try to specify cost information per design feature. As noted however, this information is usually restricted to operation costs. Especially in mass production situations, where several features are produced simultaneously in an integrated tool, it is very difficult, if not impossible, to relate complete quantitative cost information to design or process planning features.

DIDACOE does not consider features as objects for estimation, because of the difficulties involved in defining their costs. This does not imply that features are not necessary: they are still required for process planning purposes and for the generation of manufacturability information.

Although the design-by-feature approach looks attractive, the current implementation of DIDACOE does not use it. It would have been too time-consuming to develop a complete new modeller using this concept.

For the sheet metal implementation, feature recognition is used instead, based on the CAD-description as produced by the Sheet Metal Design (SMD) module of Medusa. The designer can work with this module as usual, no special actions are required.

An important question to answer was whether the feature recognition program should start from the 2D corrected blank (the unfolded sheet metal part) or from the 3D Boundary Representation model. Both representations are available from the SMD-module. It was decided to use the 2D representation for the following reasons:

- the 2D corrected blank is of central importance in the process planning for sheet metal manufacture. The 3D model does not include this information.
- Medusa offers better facilities to access 2D data than to access 3D model data. A start from 2D would probably be easier /Wie-5,7/.
- Due to the uniform sheet thickness, a lot of sheet metal parts can be conveniently represented by only one 2D view, even if they contain bends. Medusa's SMD-module offers this representation.
- It is true that 2D representations are less suitable for interpretation by computer programs. This does not imply however that it is impossible. Medusa's modeller interprets three 2D views to generate a 3D model. Swift /Swi-1/ and Kreisfeld /Kre-1/ also start from 2D CAD-representations. In addition the 2D representation used in DIDACOE will usually be generated by the SMD module from a 3D Boundary Representation model. This reduces the chance of errors.
4.6 On manufacturability information

It is difficult to relate complete detailed quantitative cost information to features or to other part characteristics that can be influenced by the designer. This increases the need for manufacturability information, for which the link to features is easier.

It will usually be possible to find some general design rules that are valid within the context of a manufacturing principle like sheet metal manufacture, injection moulding or metal removal. Most design rules, or more in general most recommendations that a process planner could give, do not depend on the manufacturing principle only however. The number of remarks that can be made will increase with the extent of detail of the process plan.

Considering this, it will usually be impossible to separate programs for the generation of manufacturability information from process planning programs. Rather, the task of process planning programs in detailed cost information tools should be extended. Next to ‘merely’ providing a process plan, a CAPP program should also generate design recommendations. This adds additional complexity to an already complex problem: the CAPP program can not just accept the design as it is, but it should also reason about possible changes in the design.

To function properly, DIDACOE should include three types of programs for the generation of manufacturability information. A general one, depending on the manufacturing principle, one associated with global process planning and one on the detailed process planning level. In addition an output program is required to present recommendation texts in relation to features in the original drawing of the corrected blank.

4.7 On neutral file formats

The current implementation of DIDACOE includes a program to recognize features in a 2D representation of a part that has been generated using the CAD-system Medusa. Datastructures and formats used in the storage of such representations differ from CAD-system to CAD-system. This implies that a new feature recognition program has to be written if DIDACOE is to be linked to another CAD-system than Medusa.

Neutral files offer the possibility to avoid this CAD-system dependency. A preprocessor linked to each specific CAD-system converts the system dependent datastructure to a commonly accepted neutral format. The feature recognition program can then take this neutral format as input. In this way the program can operate in combination with any CAD-system that can supply the neutral format. Several neutral formats are available, the most well known probably being IGES.
Although an IGES preprocessor is available in Medusa, DIDACOE does not use it (the recognition program is Medusa specific) for the following reasons:
- work is ongoing to develop one new internationally accepted standard (STEP/PDES) to replace the current collection of standards and to solve some of the problems in these standards /Ran-1,Wie-6,8/. Hence the use of IGES would only offer temporary advantages. At the start of the DIDACOE-project STEP was not completed, and even now (1989) Medusa does not provide a preprocessor for it.
- judging from the data access facilities offered by Medusa, it seemed to be easier to implement a Medusa-dependent feature recognition program than one based on the IGES-format.

4.8 On sheet metal parts

For the development of the prototype of DIDACOE, a very restricted class of sheet metal parts was selected to work on. These restrictions limit the features that are allowed in the parts as well as the manufacturing processes required to produce them.

Allowable features have been derived from the main possibilities offered by the Sheet Metal Design module of Medusa. This module allows cutlines of any shape, but non-straight line segments are restricted to second degree curves. Cutlines are used to describe the peripheries of the corrected blank (the Outer Contour) and of the holes (Hole Contours). It should be noted that SMD does not allow partial cutouts like louvers, tabs or knockouts (figure 4.5). Non-closed cutlines are allowed only when they start or end on either a Hole Contour or on the Outer Contour.

All bendlines in SMD must be straight and the material to be bent should be completely free to move. Material deformation should occur in the bendzone (of which the bendline is the centerline) only, while no double curved surfaces may be produced. In other words it should be possible to produce the feature by simple bending (forming). This excludes embossings like stiffening ribs and all other features that require some kind of (deep)drawing.

Due to lack of time it proved necessary to restrict available manufacturing processes quite heavily. Only the production from strip in progressive dies is currently (January 1990) available.

The restrictions will be defined in more detail in the following chapters.
**Figure 4.5:** The most important allowed and not-allowed features in the class of sheet metal parts selected for the DIDACOE prototype. Non-straight cutlines are restricted to second degree curves. Non-closed cutlines should start or end on a Hole Contour or on the Outer Contour. Bendlines have to be straight. No features requiring (deep)drawing are allowed.
4.9 Summary: the ideal and the implementation

A concept for a cost information tool for designers is presented in figure 4.6a. This concept reflects the ideas that were presented in the first four chapters, although passive database search for data is not shown. It will be apparent from the above that the creation of this complete DIDACOE-concept from scratch is ambitious and time-consuming. Therefore the actual implementation (January 1990) is very limited as compared to what the eventual DIDACOE should be. This actual implementation is shown in figure 4.6b.

In general DIDACOE should be able to operate in combination with different CAD-systems that deliver a 2D or 3D system specific file, a neutral format file or a feature model. Depending on the situation one of several available feature recognition modules is selected (if necessary) to provide a feature description. Design Rule Check modules and global CAPP modules per manufacturing principle, and detailed CAPP modules per type of operation supply qualitative cost information and a process plan (if required).

A Central Cost Estimation module, that includes both global and detailed methods, and operation specific time- and tool cost estimation modules provide quantitative cost information.

Qualitative information is presented in the CAD-drawing or -model, related to the features involved, to get a design-oriented feedback. For the time being the concept does not aim at providing quantitative information per feature, due to the difficulties involved in assigning costs to individual features. Consequently, this type of information is not presented in a design-oriented way. Possibilities to change this should be examined.

The current implementation always starts from a 2D Medusa-specific description of an unfolded sheet metal part. An associated feature recognition program delivers a feature description that is suitable for a selected class of sheet metal parts only.

The generation of qualitative information and the global process planning have not been implemented by lack of time. The detailed CAPP-program can be conceived as a strip layout generator.

Currently the central cost estimation module is limited to detailed estimating. Only one operation specific module for time estimation has been implemented. It handles the production of sheet metal parts from strip in progressive dies. An associated module for tool cost estimation is not included yet.
Figure 4.6a: Survey of an ideal DIDACOE. Modules marked with a * should be developed per manufacturing principle. Modules marked with a ° should be developed per operation type. CAPP= Computer Automated Process Planning. MIG= Manufacturability Information Generator.
Figure 4.6b: Actual DIDACOE-implementation in January 1990. The implementation is restricted to the use of the CAD-system Medusa and to a class of sheet metal parts. Due to lack of time, the generation of qualitative cost information and global process planning could not be implemented.
5. Detailed cost estimation concept

This chapter describes a general detailed cost estimation concept for the product, part and operation levels. It is the theoretical base for the Central Cost Estimation Module in DIDACOE.

5.1 Main Objects and the Product Structure

Usually a complex product consisting of several parts and subassem-blies will be the ultimate object to be estimated. Detailed estimation of such a product involves breaking it down into elements, estimating the costs of these elements and adding up these costs. Two types of elements are distinguished: Main Objects and Operations. The Main Objects (MO’s) that constitute a complex product are defined in a Product Structure. As illustrated in figure 5.1, three types of Main Objects appear in this structure: the Product itself (PRD), Subassemblies (SA’s) and Parts.

A Product (PRD) is a complex object, meaning that it consists of several components that have been joined by assembly operations. A product does not belong to any higher, more complex object. From the point of view of the user of DIDACOE, it is the ultimate objective of the design and manufacturing process. Typical examples include all products that are sold to consumers, like cars, telephones and mixers.

A Subassembly (SA) is also a complex object. The only difference with a product is, that it is intended to belong to a higher, more complex object (a product or another subassembly). A SA is a joined collection of Main Objects, for which it is useful in some way to distinguish it as a unit within a more complex Main Object. The definition of a SA will usually depend on the assembly process for this last object. Typical examples include the motor within a car or mixer and the receiver of the telephone.

Parts or components are simple Main Objects, meaning that there are no other Main Objects belonging to them. They are intended to belong to a higher, more complex object. Parts come in two types that will be called MakeParts and BuyParts.

A MakePart (MP) is a component that the user of DIDACOE intends to manufacture himself. Per definition, a MakePart can be produced without using assembly operations.

A BuyPart (BP) is an element that the user of DIDACOE does not intend to manufacture himself: he will buy it. BuyParts can be complex, but if they are, their elements will not be distinguished. From the point of view of DIDACOE they are always simple. Typical examples are standard parts like bolts, but more complex objects like bearings or motors, that are bought as a unit, can also be considered as BuyParts.

It should be noted that the same object can be defined in different ways by different users. Motors or bearings for example will often be conceived as
Figure 5.1: Example of a product structure. The structure defines all Main Objects (subassemblies and parts) that appear in a product. In addition the Direct Number of Appearances of a 'lower' object in a 'higher' object is specified.

BuyParts or, if the user intends to make them himself, as subassemblies. Manufacturers that specialize in these objects will conceive them as products however.

At first sight, relations between Main Objects in the product structure are simple. They can be described as 'part-of' or 'has-parts' relations. There are some complications however. The same component or subassembly can appear in a more complex Main Object more than once. In this case two situations should be distinguished.

First, an object can appear directly in a higher object more than once (example: four times Part2 in the product, or two times Part2 in SA1 in figure 5.1). This situation is dealt with by specifying a Direct Number of Appearances (DNA) for every relation between two MO's. This DNA is important when costs of all elements of an object have to be added up to get the object costs.

Second, an object may appear several times within the same product, but on different levels in the product structure (example: Part2 appears in PRD, SA1, SA3 and SA4 in figure 5.1). Associated with this situation is a Total Number of Appearances (TNA) of a part in a product. If necessary, this TNA can be extended to include appearances of the same object in different products. The TNA is important for the calculation of the quantity of objects that has to be made or bought. Note that the TNA's can be calculated when all relations between Main Objects and their associated DNA's are specified.

5.2 Operations and additional objects in the Production Path

The product structure contains only part of the information necessary for an estimate. Costs of a product or subassembly are not equal to the sum of the costs of the Main Objects that directly belong to them. Costs of the assembly operation have to be added and maybe the costs of additional operations as well. Moreover it is necessary to distinguish operations on part-level to estimate costs for a MakePart. In general, a sequence of several different
operations will be necessary to produce a Main Object. These operations and their sequence are defined in a Production Path as illustrated in figure 5.2. Such a path has to be defined for every Main Object in the product structure, except for BuyParts, for which operations cannot be specified. The output of the last operation in a path will always be the Main Object to which the path belongs.

On product or subassembly level the input for the first operation in the sequence will be a Set of Main Objects (at least two) as defined by the product structure. This operation is an Assembly Operation per definition, and it is the only one that is allowed to be an assembly operation.

On part level the input for the first operation will be the purchased material from which production starts. This material is represented by one or more Basic Objects (BO’s). Even if several BO’s are present, this operation is never called an assembly operation however. Between each two operations in a path temporary objects can be defined that represent unfinished states of a Main Object. Borrowing terminology from van 't Erve /Erv-1/ these states will be called Less Worked Objects (LWO’s).

Per definition every production path is straight and simple: it does not have any ‘branches’. Any operation in the path is supposed to provide one type of output object only. Operations that take more than one type of object as input are allowed as first operations in a path only.

![Diagram of Production Paths](image)

**Figure 5.2:** Examples of Production Paths on Part-level and on Product- or SubAssembly-level. These Paths specify the operations that are necessary to produce a Main Object and their sequence. Basic Objects form the input on Part-level. MakeParts, BuyParts and SubAssemblies are the input on higher levels. Less Worked Objects (LWO) are intermediate objects that represent an unfinished state of a Main Object.
Basic Objects can only exist in relation to a Main Object. In addition they differ from BuyParts in intended use. BO's will always be used as input for the first transformation operation on part level, whereas BuyParts enter directly in an assembly operation on Product or Subassembly level.

Typical examples of Basic Objects include a sheet, a strip, a bar or inserts used during injection moulding. More abstract objects like a 'quantity of granulate' are also considered as BO's however. Two types of Basic Objects will be distinguished: physical objects and non-physical objects. Physical objects are really existing solid objects. Non-physical objects are abstract objects that can not clearly be defined as one solid. Examples are a 'quantity of granulate', a 'quantity of resin' or a 'quantity of melted metal'. Some objects like an 'area of glass fibre fabric' may fit into both categories.

Similar to the definition of DNA's for Main Objects in the product structure, Basic Object Frequencies (BOF's) specify the theoretical number of BO's required for one out-object of the first operation on part level. For non-physical objects this BOF is not defined.

The Less Worked Objects to be distinguished depend directly on the definitions of the operations. Consider for instance the production of a sheet metal part. Using a compound die such a part can be produced by one stroke of a press. Obviously this is one operation and there are no LWO's. Using a progressive die with several steps the same part can be produced from coil by several strokes of the press. In this case LWO's could be defined as the states of the Main Object after each step in the die, but this does not seem to be very convenient. The use of a set of conventional dies on separate presses will lead to a clear distinction of several operations with clearly defined intermediate objects.

The problem of defining operations can not be solved in general. Due to the large variety in types of operations, a convenient definition has to be made for every type separately.

From the point of view of the general detailed cost estimation concept, the specification of the production path can be very abstract. All that is needed is the number of operations, and on part level the number of Basic Object(s), their type (physical, non-physical) and their BOF. This already implies the existence of LWO's.

5.3 A black box model for operations

To enable the development of a general detailed cost estimation concept, it is necessary to define a general model for operations. The first part of this model, called the 'black box model' is introduced here. It considers the material input and output for an operation as shown in figure 5.3.

The model considers two inputs: the Input (with capital I) and Additions. The Input represents the main material to be processed and has to be present for any operation. Additions represent small amounts of material that are added to the Input and that will appear in the Accepted Output. They will not be present for every operation. Typical examples of Additions are adhesives, welding material, paint and other surface layer materials. In contrast, fuel for the
machines and materials that serve a smooth processing only, like coolants and lubricants, are not considered to be Additions as they will not appear in the Accepted Output. The model does not consider them.

Four types of output are distinguished. The Accepted Output represents the main intended material output that has to be present for every operation. The Rejected Output represents objects that come close to those in the Accepted Output, but that are not accepted during quality control or that are rejected right away because of obvious defects like tears, wrinkles, voids or incompleteness.

Scrap represents input-material that has been removed or that did not reach the Output for other reasons. Spillage represents additions-material that did not reach the Output. Note that Rejected Output, Scrap and Spillage need not be present for every operation.

The Output of an operation is defined as the sum of Accepted Output and  

![Diagram](image)

**Figure 5.3:** Black box model for an operation. The Input is the main material input for an operation. An Addition is a small amount of material that is added during the operation such as adhesives or paint. Part of the intended output is Accepted, another part is Rejected because of defects. Scrap is removed or unused Input-material. Spillage is unused Addition-material. Input and Output are conceived to consist of discrete objects which permits an Operation Multiplication Factor and a Rejection Factor to be defined. Operations are allowed to produce one type of accepted output-object only.
Rejected Output. This Output potentially is a mixture of Input-materials and Addition-materials. The sum of Rejected Output and Scrap will here be called Waste.

Both the Input and the Output are conceived to consist of discrete Objects: the Main Objects, Less Worked Objects and Basic Objects from the production path. The model assumes that the Output always consists of real physical solids and that all these objects are of the same type. This excludes for example the production of several different parts from strip or sheet for reasons of material use optimization.

Several different types of objects are allowed in the Input, but only if the operation is the first operation in a production path. On part level these objects are also allowed to be non-physical objects, as described for Basic Objects above. Even in these cases it will be possible to define discrete objects. Operations like injection moulding and casting show a clear cycle that repeats itself: one stroke of the injection moulding machine for instance. In this case the ‘quantity of material’ used during one cycle can be used as Input-object. For continuous operations that do not show a clear repeated cycle, like extrusion, discrete objects can be obtained by considering the production in some unit of time or by assuming some unit of Output and considering the associated unit of Input.

The Additions can also be conceived as discrete Objects. They will be called Added Material Objects (AMO’s). For the time being only one AMO can be associated with an operation. Although possible, Scrap and Spillage are not modelled as Objects.

The separation into discrete objects permits the definition of an Operation Multiplication Factor (OMF) and of a Rejection Factor (RJF).

In principle the Operation Multiplication Factor is defined as the number of Output-Objects that is produced from one Input-Object:

$$\text{OMF} = \frac{Q_{out}}{Q_{in}} \quad \text{or} \quad Q_{out} = \text{OMF} \times Q_{in}$$

The Rejection Factor (RJF) is the fraction of the Rejected Output-Objects in the total of Output-Objects. Using this factor the number of Accepted Output-Objects (ok) can be calculated from the number of Input-Objects:

$$Q_{ok} = Q_{out} - Q_{out} \times \text{RJF} = Q_{in} \times \text{OMF} \times (1 - \text{RJF})$$

For operations that take several types of objects as Input the OMF as defined above has no clear meaning. For assembly operations the OMF is not used. The quantity relations can be found using the DNA for each Input-object:

$$Q_{out} = \frac{Q_{inMO}}{\text{DNA}_{MO}} \quad \text{(for each of the input-objects separately)}$$

$$Q_{ok} = \left(\frac{Q_{inMO}}{\text{DNA}_{MO}}\right) \times (1 - \text{RJF}_{op})$$
For first operations on part level that use Basic Objects of different types, the use of the OMF depends on the type of Basic Object. For physical objects the situation is exactly the same as shown for assembly operations, but now the DNA is replaced by the Basic Object Frequency:

\[ Q_{ok} = (Q_{\text{BO}}/BOF_{\text{BO}}) \times (1-RJF_{\text{op}}) \]  
(for each physical Basic Object)

For non-physical objects like a 'quantity of granulate', the OMF is defined as normal. In these cases the OMF will usually be equal to the multiplicity of the die. This difference between physical and non-physical BO's can be understood by looking at the production of an injection moulded part that contains two inserts. In the case of single production two types of Basic Objects will be defined: two physical objects 'insert' with BOF=2 and one non-physical object 'granulate'. The associated OMF is 1. The simultaneous production of four parts requires eight 'insert'-objects with BOF=2 but still one object 'granulate'. The OMF is now 4.

5.4 Quantities

5.4.1 Total Quantity and Batch Quantity for Main Objects

The Total Quantity (TQ) for a Main Object is the expected total number of objects that will be produced over the years (the ultimate production quantity). For a complete product it will probably be derived from market research in some way. Total Quantities are necessary to determine the initial costs (such as tool costs) per object.

The Batch Quantity (BQ, the economic order quantity) is the number of objects that will be made in one batch. Machines and tools have to be prepared once for this quantity. As a consequence it is used in relation to set-up costs. The Batch Quantity will usually be chosen as economical as possible, considering manufacturing costs, storage costs and investments in materials for example. In practice, the BQ may change over the years. For estimation purposes it is assumed to be constant however.

For a product, the total quantity should be specified in the design assignment. In addition it is assumed that the designer knows the batch quantity for the product.

Quantities for subassemblies and parts could be derived from those for the product using the product structure information. It does not seem right however to make SA and Part quantities directly dependent on product quantities. Subassemblies and parts may appear in different types of products (or product variants), which implies that their quantities can not be derived by looking at the TQ, BQ and structure for just one product. In addition, even if a part appears in only one product, there is no reason for its batch quantity to be directly related to the product BQ.

So, on the one hand, SA and Part quantities are related to those for the product, but on the other hand they are independent to a considerable extent. To solve this problem the following approach was adopted.
A total quantity and a batch quantity are defined for a product. Both values have to be supplied by the user of DIDACOE. For subassemblies and parts the user can make an estimate without having to specify the complete product structure first. This estimate will be based on Original Quantities (ORTQ, ORBQ), to be supplied by the user. When the SA or the Part is used as an element in a more complex Main Object, an estimate for this MO will cause a re-estimation of the SA or Part based on Active Quantities (ACTQ, ACBQ). These active quantities can be influenced by defining Quantity Sources (TQS, BQS).

The Total Quantity Source (TQS) can have one of three values:
1- 'fixed', indicates that the original total quantity will be used,
2- 'sumall', will cause the ACTQ to be derived by taking into account the ACTQ's of all objects to which the SA or Part belongs,
3- 'sumcur', is the same as 'sumall' but limited to appearances of the SA or Part in the Main Object currently being estimated.

The Batch Quantity Source (BQS) is specified by a number between 0 and 100. Zero indicates that the ACBQ will be the same as the ORBQ (fixed). In all other cases the ACBQ will be computed as a percentage (indicated by the number) of the ACTQ.

54.2 Quantities in a Production Path

The preceding paragraph specifies how quantities for Main Objects are determined. These quantities will be used to derive the quantities for objects and operations in the production paths. Due to the rejection factors and multiplication factors of the operations, the latter quantities will not be equal to the Main Object quantities.

Considering one operation (op) the total number of out-objects and in-objects can be calculated from the number of accepted out-objects (ok) using the equations from paragraph 5.3, for example:

\[ Q_{in} = Q_{ok} / (OMF_{op} \times (1-RJF_{op})) \]
\[ Q_{out} = Q_{in} \times OMF = Q_{ok} / (1-RJF_{op}) = Q_{op} \]

These equations (or the similar equations with DNA or BOF) can be used both for the batch quantity and for the total quantity.

Rejection will have a cumulative effect in production paths that consist of more than one operation. If the last operation in a sequence causes rejection not only this operation has to be executed an additional number of times to produce the required accepted output, but all preceding operations as well. This implies that we have to calculate backwards from the known batch and total quantities for the Main Object (the accepted output of the last operation), using rejection and multiplication factors of all operations in the production path, in order to find the total and batch quantities for operations, Less Worked Objects and Basic Objects.
It will be clear from the above, that an Operation Total Quantity (OTQ) and an Operation Batch Quantity (OBQ) can be defined. These quantities are per definition equal to the total number of out-objects produced, including the rejected ones. They do not always indicate the required number of operation cycles, as more than one out-object can be produced per cycle.

5.5 Types of costs

The detailed cost estimation concept distinguishes four types of costs: material costs, operation costs, tool costs and other costs.

The other costs are undefined, meaning that they can be used for any purpose. Typically they could be used to specify costs for research and development, marketing costs or distribution costs. Whether other costs are necessary or not depends to a large extent on the definitions of the machine and operator rates and on the precise definition of the concept 'manufacturing costs'. The rates may already take into account research and development costs for example. If they do not, it will depend on the definition of manufacturing costs whether they should be included using other costs or not. The general concept presented here is quite flexible from this point of view. Other costs can only be specified for Main Objects by supplying the costs and the quantity of objects for which these costs are made.

Operation costs are a type of cost. Therefore they do per definition not include tool costs or material costs, although these latter types of costs can be associated with an element 'operation' in the production path.

For the element 'operation' three types of costs are distinguished: material costs (from the Added Material Objects), tool costs and operation costs. The term 'integrated operation costs' is used to indicate the total costs for the element 'operation', which includes all types of costs.

5.6 Estimation of material costs

5.6.1 Introduction

Manufacturing a Product includes the following input of materials:
- Basic Objects (BO's), that represent the main materials from which the production of MakeParts starts. Typical examples include a quantity of granulate, a sheet metal strip or a metal bar,
- BuyParts (BP's), that represent simple or complex elements that are bought and that enter directly in an assembly operation. Typical examples include bolts, motors, electrical wires or stickers,
- Added Material Objects (AMO's), that represent small amounts of material that are added to the main material by an operation. Typical examples include paint, adhesives or welding material.

In addition there are materials that serve a smooth processing only, like lubricants or coolants. They are not considered when estimating material costs. Costs for these materials are supposed to be included in the machine rate.
The expenditures that are necessary to buy BO's, BP's and AMO's will be considered as gross direct material costs. Indirect material costs associated with the purchase, storage and transport of materials will be taken into account using a percentage of the direct costs.

Material output in product manufacturing includes:
- Main Objects (Product, Subassemblies and MakeParts),
- Waste (rejected Main Objects, rejected Less Worked Objects, Scrap),
- Spillage (wasted AMO-material).

It is assumed that waste can be sold again to provide a Waste Revenue that is subtracted from the gross direct material costs to provide the net direct material costs. Spillage is neglected.

5.6.2 Costs of Basic Objects, BuyParts and Added Material Objects

The costs of Basic Objects, BuyParts and Added Material Objects form the base for the material cost estimate. For all three types of objects direct costs are calculated per batch using:

Direct Cost per Batch = Cost per Piece * Active Batch Quantity

For BuyParts the batch quantity is determined using the original batch quantity or the batch quantity source as explained in 5.4.1.
For Basic Objects the batch quantity is derived from the active MakePart quantity as explained in 5.4.2. This implies that BO quantities depend directly on the quantities of the MakePart for which the BO is defined. Independency of BO quantities similar to that for MO quantities is ignored for the time being.

The AMO batch quantity is assumed to be equal to the operation batch quantity for the operation to which the AMO belongs. This is a temporary solution.

The 'Cost per piece' can be determined using one of three available methods:

- the 'quantity method' implies a specification of the cost per quantity of objects:
  Cost per Piece = Cost per Quantity / Quantity
  This method will especially be convenient for BuyParts. The specification per quantity (instead of a direct specification per piece) has been chosen for ease of use. Prices of small standard BuyParts are often available per package of 24, 100, 1000 or the like.

- the 'unit method' uses a specification of cost per unit. The units mentioned can be cubic metre, square metre, metre, litre or any other suitable unit. Costs per Piece are calculated from:
  Cost per Piece = Cost per Unit * Number of Units per Object
  The 'number of units' represent the volume, area, length and the like of the Object considered.
- The 'kilo method' is a special case of the previous approach, where the unit used is kilo. The 'number of units' is the Object Mass, calculated from:
  \[ \text{Object Mass} = \text{Object Volume} \times \text{Specific Mass of material} \]
  The kilo method is distinguished separately because it is applied often and because its use is recommended for Basic Objects. Use of this method facilitates the calculation of Waste Revenue.

Indirect costs are calculated for all three types of objects using an Indirect Material Factor (IMF). This factor is assumed to be defined centrally for the whole DIDACOE and defines the indirect costs as part of the direct costs:

\[ \text{Indirect Cost per Batch} = \text{Indirect Material Factor} \times \text{Direct Cost per Batch} \]

Gross material costs for are the sum of direct and indirect costs.

5.6.3 Estimation of MakePart material costs

MakePart material costs per batch can be calculated by adding the gross material costs per batch for all Basic and Added Material Objects in the production path. The costs per MakePart (per piece) can then easily be derived by dividing this sum by the active batch quantity for the MP.

This approach does not consider that waste may be sold or reused. The detailed cost estimation concept takes into account the Waste Revenue under some conditions. Waste Revenue is computed per batch of MakeParts and subtracted from the above sum of gross material costs per batch to get the net material costs.

As illustrated in figure 5.4 the calculation of the amount of waste can be approached in two ways. The first approach considers the waste produced by every operation in the production path. This waste consists of rejected out-objects and removed or unused material (the scrap). The total waste can be found by adding the contributions of all operations. In the second approach, the path is conceived as a black box and only its input and output are considered. The difference between the two is the amount of waste. Obviously, the last approach is much simpler.

The only convenient way to calculate Waste Revenue is to express the total amount of waste in kilo and to multiply this Waste Mass by a Waste Price per Kilo.

If the kilo method was chosen to estimate Basic Object costs, the calculation of the Waste Revenue per Batch is simple:

\[ \text{Waste Volume} = \text{BasicObject Volume} - \text{MakePart Volume} \]
\[ \text{Waste Mass} = \text{Waste Volume} \times \text{Specific Mass} \times 10^{-6} \]
\[ \text{Waste Revenue} = \text{Waste Mass} \times \text{Waste Price per Kilo} \]

(Volumes in mm$^3$ per batch, mass in kg per batch, specific mass in kg/dm$^3$ or g/cm$^3$)
If Basic Object costs were defined using the quantity method or the unit method, the calculation of the waste mass may pose problems due to the lack of material data or the lack of the BO Volume. Of course, the required data could be derived after all, but then it would have been better to use the kilo method straight away. Therefore Waste Revenue will only be calculated if the kilo method has been used.

The above procedure also presents problems when several Basic Objects of different type have been defined. A temporary solution has been implemented for the case in which costs for all BO's have been defined using the kilo method. In that case the MakePart Volume is split in as many parts as there are BO's. Each part is taken equal to the share of the volume of the BO involved in the total volume of the BO's together (thus assuming equal waste percentages for all BO's). The calculation of the Waste Revenue can now be applied for each BO separately. Note however that the waste produced may be a mixture of the different materials, which may be hard to sell or reuse.

**5.6.4 Estimation of Product material costs**

The estimation of material costs for products and subassemblies has been simplified. Two important assumptions have been made to avoid complex problems.

First, waste revenue on these levels is completely neglected. Waste always consists of a mixture of materials here, and the method applied for multiple Basic Objects will usually not be applicable here due to lack of information on BuyParts.

Second, all operations on these levels are assumed to have a rejection factor of zero (no rejection). Without such an assumption the costs of a MakePart on
product or subassembly level could have different values, because different 'paths' in the product structure could cause different 'rejection costs'.
With the two assumptions, material costs for products and subassemblies can easily be calculated by adding the material costs per piece of all Main Objects directly belonging to them, taking into account the DNA's, and by adding the costs of AMO's appearing in the production path.

5.7 The grey box model for operations

Opening the black box presented in paragraph 5.3 for just a moment one can get a glimpse of its contents without seeing all the details. This permits the description of what is called the 'grey box model' for operations. In this model an operation is conceived as an action or as a collection of related actions that are performed in some useful order. These actions are performed by an operator and a machine using a tool and they convert Input and Additions into Output. Waste and Spillage. Operation times are associated with the actions while rates are associated with machines and operators (figure 5.5).

Figure 5.5: The grey box model for operations. An operation is an action or a collection of potential actions. Three types of actions with their associated types of time can be distinguished: preparative, secondary and main. An operation is performed by an operator and a machine using a tool. Rates are machine- and operator-costs per hour. Rates and times lead to operation costs. Other types of costs associated with an operation are tool costs and material costs (from an Addition).
5.7.1 Actions and Times

Three types of actions are distinguished with three associated types of operation times:

Preparative Actions are unproductive actions that are needed once per batch. Typically these actions are set-up actions required to install and test the tool and the machine. In a broader view, studying the assignment and gathering necessary materials and tools could be conceived as preparative actions. This type of action leads to Preparation Time (PT). The PT is per definition independent of the operation batch quantity. It is specified per OBQ, not per out-object.

Secondary Actions are unproductive actions that are performed more than once per batch. Often they will be performed once per out-object, but there are also actions that are performed once in a while for a number of out-objects together. Examples include moving a cutting tool from the initial position to the position where actual cutting starts, fixing and unfixing of objects, putting a new strip in the press, removing finished products from a press or the opening and closing of an injection moulding machine. The associated Secondary Time (ST) is specified per out-object. This implies that times for actions that are performed for a number of out-objects together have to be divided by this number to define the ST.

Main Actions are the real productive actions that change the input state into the output state. The actions can be performed per out-object or for a number of out-objects together. Examples are the actual cutting on a lathe, injection and cooling during moulding or a stroke of the press during stamping. The associated Main Time (MT) is specified per out-object. This implies that times for actions that produce a number of out-objects simultaneously have to be divided by this number to define the MT.

The definition of an operation includes the definition of operation limits and the definition of all actions and their type. A number of general remarks can be made related to this topic:

- The detailed cost estimation concept more or less assumes that Input, Additions and Tools are present near the machine at the start of the operation. The transport of these materials from storage to the working place is not conceived to be part of the operation. The same is assumed for transport of accepted output and scrap to storage, to the next working place or to the central dustbin. Costs for internal material transport are supposed to be covered by the indirect material costs. This more or less defines the limits for every operation. Nevertheless, the user of the concept is free to define the operation limits and the indirect material factor as desired.

- The precise definition of actions will depend on the type of operation and on personal preferences. In principle the user is completely free to consider actions as secondary or main. In addition, actions can be defined globally or in detail. The global action 'put object in tool' could be replaced by the
sequence of detailed actions: grasp object, lift object, move object to tool, position object in tool, fix object. These distinctions are not considered by the general estimation concept but by operation specific concepts.

It is probable that a lot of operations can not be defined as a fixed set of actions performed in a fixed sequence. Often a collection of potential actions will be defined with a set of conditions specifying when they will be used and in which order. The action 'remove out-object from press' for example is not required when out-objects fall away automatically through a hole in the press table. An action like 'operate side-stops' is only meaningful if there are side-stops. Note that it would not be convenient to define a separate operation for every new set or every new sequence of actions.

The estimation concept assumes that action times are normal or standard times /Wil-1,Mal-2/. These times are valid for average employees in normal production situations or they are 'target' times that can be realized in some ideal, desired situation. For the practical situation several allowances have to be applied to them. Allowances exist for personal care, breakdowns or other delays, and tiredness. In DIDACOE these allowances are combined in the Time Allowance Factor (TAF). For flexibility purposes different TAF's can be defined for each of the three types of time.

Different actions may require a different use of production means. One, two or more operators may be involved in installing the tool and testing the set-up. In the case of automated production, one operator may control several machines at the same time. This implies that action times may be different for operator and machine. These differences are reflected by the Human Attention Factor (HAF) and the Machine Use Factor (MUF). Different HAF's and MUF's can be defined for the three different types of time. Usually either machine times or operator times will be estimated. Using the HAF's, operator times can be derived from machine times.

The MUF's allow the opposite.

Note that TAF's, HAF's and MUF's can not be specified per action but only per type of action. This is due to their inclusion in the general estimation concept, which only distinguishes types of actions.

5.7.2 Machines, Operators and Rates

The term 'machine' is used here in the broadest possible sense. A complete 'working place' is also called a machine. In stamping from coil for example, the press, the strip straightener, automatic feeding equipment, spoolers and despoolers and potentially present 'take-out-and-stack' equipment are together conceived as one machine. This justifies the fact that only one machine can be associated with an operation. Costs of this machine are taken into account by specifying a Machine Rate, which are costs per hour for the machine. Obviously, it is not always necessary to associate a machine with an operation.

The operation definition should specify at least if a machine will be used. If so, at least the type of machine should be defined. Rates are defined for specific
machines only. This implies that a machine selection is required if several machines of the suitable type are present in the company.

Operators are human beings that perform an action or that supervise or control actions performed by the machine. In the concept, it is not necessary to associate an operator with an operation. The operation definition should specify if an operator is required. If so, an Operator Level has to be defined. Directly related with each level is an Operator Rate: the costs per hour for operators of this level. It will be obvious that, in contrast with machines, rates are not specified for specific operators.

5.8 Estimation of operation costs

5.8.1 Determination of operation times

Estimation of the operation times and the determination of the associated time allowance factors is one of the most difficult and most important tasks in detailed cost estimation. In general two approaches can be distinguished. Operator dependent action times can be measured in practice or calculated in some way. Methods like Time Study Analysis, Motion Time Analysis, the Work Factor System and Methods Time Measurement have been developed for this purpose (/Wi-1,Mal-2/).

Machine dependent action times can usually be calculated based on the machine parameters that are suitable for the operation. Examples of such parameters are the revolutions per minute and translation speed for a lathe or the number of strokes per minute for a press or injection moulding machine. The general estimation concept does not consider the determination of times as this is typically an operation specific topic.

5.8.2 Determination of machine and operator rates

DIDACOE does not consider the determination of rates. The way in which these basic data have been determined and their precise definition is of no importance to the concept. Yet one remark is necessary. In principle rates are costs per hour for a machine or an operator. Rates will probably be used however to assign all kinds of indirect costs to the products. Costs of buildings, energy, coolants, lubricants, standard tools, heating, cleaning and the like will typically be included in the machine rates. Costs for research and development might be taken into account using the operator rates. The user of the detailed estimation concept is completely free to define the rates. This implies that it is the user's responsibility to ensure that the eventual cost estimate is complete and correct. Rates, indirect material factor, tool costs, operation limits and other costs should be defined in such a way that all costs are taken into account and that nothing is counted double! This depends to a large extent on the definition of the rates.
5.8.3 Operation costs for a MakePart

With the definitions explained earlier, operation costs may be estimated using the following equations, where index 'p' stands for preparation, index 's' for secondary and index 'm' for main:

Machine Time per Batch =
Preparation Time * MUF_p * TAF_p +
Secondary Time * MUF_s * TAF_s * Operation Batch Quantity +
Main Time * MUF_m * TAF_m * Operation Batch Quantity

Machine Cost per Batch = Machine Time per Batch * Machine Rate

Operator Time per Batch =
Preparation Time * HAF_p * TAF_p +
Secondary Time * HAF_s * TAF_s * Operation Batch Quantity +
Main Time * HAF_m * TAF_m * Operation Batch Quantity

Operator Cost per Batch = Operator Time per Batch * Operator Rate

Operation Cost per Batch =
Machine Cost per Batch + Operator Cost per Batch

Operation costs for a MakePart can now be calculated by adding costs per batch for all operations in its production path and by dividing this sum by the active batch quantity for the part.

5.9 Estimation of tool costs

Small commonly used tools like hammers and spanners, and tools that are used for several products like drills, standard punching tools and cutters are usually not considered when discussing tool costs. These tools can be conceived to belong to the 'shop floor' in general or to be 'part' of the machine. Hence their costs should be included in the machine rate in some way. Tool cost estimating focuses on special made tools that will be used for the production of one type of object only. For the time being only one tool can be associated with an operation. Of course it is not necessary to specify a tool for an operation. The operation definition should specify whether a tool is used or not, and if so, its type.

The detailed cost estimation concept uses the following information associated with tool costs.

Basic Tool Costs are the costs to make or buy one tool. The way in which these costs are estimated will depend on the type of tool (the type of operation). The method of estimation can only be detailed for a specific operation. Hence it is a task for the specific operation modules to calculate these costs.
Tool costs are considered to be initial cost. Investments have to be made in tools before production can start. This investment should be considered to be a loan (either internally or externally) and thus interest will have to be paid. Interest Cost can be calculated in many different ways with the choice depending on company policy /War-1/. For the time being only one calculation method is provided. Interest Costs are determined using:

\[
\text{Interest Cost} = \left( \text{Basic Tool Cost} \times \text{Interest Factor} \times \text{Interest Time} \right) / 2
\]

The Interest Factor is the part of the Basic Tool Costs that has to be paid as interest every year. The factor is assumed to be defined centrally for the whole DIDACOE. The Interest Time is the period in years during which interest has to be paid.

Repair and Maintenance Costs are assumed to be defined as a percentage of Basic Tool Costs.

The Number of Tools Required depends on the Operation Total Quantity and on the total number of objects that can be made with one tool. This last number is defined as the Tool Life, which is not necessarily equal to the associated number of operation cycles, as more than one out-object can be produced in every cycle. As the OTQ is defined as the total number of out-objects (including the rejected ones), Tool Life and OTQ are directly comparable:

Number of Tools Required = \frac{\text{Operation Total Quantity}}{\text{Tool Life}}

Theoretically fractions should always be rounded up. In addition it should be possible to specify a Number of BackUp Tools, to be added to the above result.

Total Tool costs per operation are now calculated from:

Total Tool Cost =

\[
(\text{Basic Tool Cost} + \text{Interest Cost} + \text{Repair & Maintenance Cost}) \times \text{Number of Tools Required}
\]

Tool costs per MakePart can be found by adding tool costs for all operations in its production path and by dividing this sum by the active total quantity for the part.
5.10 Information Summary

Figure 5.6 clarifies the required input for a detailed cost estimate on part level, and the way in which this information is used. Input data can be recognized by a thick line under their box and by the lack of a connection on the bottom side of the box. In practice this input may be taken from a CAD-system, from databases or from operation specific time or tool cost estimation modules. In these cases additional input may be required, such as a material identification that serves as a keyword to access a datafile. This type of information and some administrative data are not shown. Additional information on the data involved in the cost estimation concept can be found in Appendix 1.

![Diagram](image)

**Figure 5.6: Information summary for detailed cost estimation concept**
Figure 5.6 (continued)
Figure 5.6 (continued)
6 Cost estimation for a class of sheet metal manufacture

An important part of the input for the central cost estimation module, as summarized in 5.10, can be supplied by operation specific modules. Such modules should supply the following information:
- rejection factor and multiplication factor,
- machine rate and machine use factors,
- operator level and human attention factors,
- basic tool costs, interest time, repair and maintenance percentage, tool life and number of backup tools,
- if an AMO is used, its estimation type and the associated data required to estimate its costs,
- preparation time, secondary time, main time and associated time allowance factors.

Specific modules will be developed for types of operations. The central cost estimation module will check the types of the operations in the production paths. Whenever it recognizes that a specific module exists for a type it will call this module. For all other types of operations the user will be prompted to supply the input.

In this chapter the operation type SMT: 'production of sheet metal parts from Strip using progressive dies' will be defined. It is shown, or at least discussed, how a specific module for this type of operation can provide the output listed above.

6.1 General description of the type of operation

The production of sheet metal parts from strip using progressive dies is a stamping process. It is hard to define what stamping of sheet metal actually is, but according to Stein (in /Bra-1/) it includes cutting or shearing, bending or forming, and drawing or deepdrawing operations. Cutting around the periphery of a part is called 'blanking'. Cutting holes in a workpiece is called 'punching' or 'piercing' (figure 6.1). In 'forming', the operation produces one or more plane faces which are at an angle to the original flat plane of the blank.

These operations can be performed in many different ways (see also paragraph 10.3). Typically, the size and shape of the completed workpiece are determined by a die set consisting of two halves, between which the sheet metal is placed. The upper half (the punch) is attached to the ram of a press and moves up and down, while the lower half (the die, for example the cutting plate) is fixed on the table of a press. This press provides the force needed to effect the change.

A progressive die is a complex die set in which many punches and dies for partial stamping operations are combined. This complex die can be conceived to consist of 'steps' which correspond with zero, one or more partial
operations. A strip is moved through the die set in such a way that a specified area of this strip lies under the next step in the die just before each next stroke of the press. Thus, after a number of strokes equal to the number of steps, every partial operation present in the progressive die has been performed on the specified area of sheet metal. This principle is illustrated in figures 6.2 and 6.3.

More detailed information on the manufacturing of sheet metal parts in progressive dies can be found in /Her-2,Rom-1,Bre-1,Gei-1,Wie-2,5/.

Figure 6.1: Three stages in a typical 'piercing' or 'blanking' operation (Source: /Bra-1/).

Figure 6.2: Production of two parts per stroke in a progressive die with four steps. The first step pierces the holes, which are used for positioning using pilot pins as well. The second step produces the notches, the third step contains the forming operation and the final step cuts off the two parts. (Source: /Her-2/)
6.2 Operation type model

6.2.1 Input and output

Operations of type SM1 do not use Additions. The Input in the sense of the 'black box model' is a Strip. This is a rectangular sheet metal object of constant thickness, for which the long edges are considerably longer than the short edges. Edges are assumed to be straight, without any notches or the like. The material inside the rectangle is flat and without holes. The length of strips is limited by the fact that they should be reasonably easy to handle by an operator. This distinguishes them from the much longer coiled strips that require special feeding equipment to be processed. Typically strip length will vary from 1 to 3 meters.

In practice the maximum strip width is determined by the dimensions of the largest available press. The maximum width for which the cost estimation data presented here are valid is 375 mm. Strip thickness will typically range from 1 to 6 mm, although values outside this range are possible. Strips can be bought (as physical Basic Objects) or they can be made, usually by shearing them from sheets.

The Output of the operation consists of sheet metal parts, which can have a variety of shapes and dimensions. Usually however the parts produced will be relatively small. From a cost estimation point of view the features that may appear in the parts are hardly restricted. The limitations considered in paragraph 4.8 do not apply to the cost estimation module itself. It may be however that time data presented here are not valid for parts containing bends. This fact is neglected.

The operations will almost always produce Scrap, consisting of punched out hole-material and of the remainder of the Strip.
By lack of additional information the Rejection Factor for this type of operation has been fixed on 0.02 (two percent). For the time being this factor does not depend on the circumstances. The Operation Multiplication Factor is the number of parts that is produced from one strip. Its calculation is presented in paragraph 6.4.

6.2.2 Actions

At the start of the operation the tool and the strips are assumed to be ready for use near the press. Strips are assumed to be greased. At the end of the operation accepted parts (still greased) and waste are left near the press, ready for transport. This implies that transport of tool and materials to and from the press environment is not considered part of the operation.

Two preparative actions are distinguished that will always be necessary:
- prepare the press, install the tool and test the set-up.
- remove tool from press and clean both.

Nine potential secondary actions will be considered:
- pick up a new strip and position it in the tool in such a way that the set-up is ready for the first stroke.
- move strip over prescribed distance after each stroke of the press.
- operate side stops. This action is required only if side stops are used in the tool as an aid to position the strip during the first strokes of the press. If they are, the action may be required several times for each strip.
- take used strip out of tool, turn it and position it in the tool ready for the first stroke again. This action is meaningful only during so called Turn Cutting (see /Pic-2/).
- take used strip out of the tool and put it aside. This action is almost always necessary, although the strip remainder may be very small.
- take manufactured part out of the tool and put it aside. This is necessary in special cases only. Normally, parts are assumed to be removed from the tool automatically in some way.
- take scrap out of the tool and put it aside. This action is ignored during time estimation. Scrap is assumed to fall away through the hole in the press table or to be removed in some other way.
- operate press. This includes switching the press on and off, handling the safety screen and waiting for the start of the stroke of the press.
- other secondary actions. These represent actions that are required once in a while like sorting the output, making piles of out-objects, counting the parts and lubrication of the press.

Only one main action is distinguished: the stroke of the press.

For all three types of action the Time Allowance Factor is fixed on 1.25, independent of the circumstances.
6.2.3 Machine

One machine is used for the operation. This machine is always of the type 'press'. For the time being this implies a single working mechanical press. Hydraulic presses are not covered yet (see 6.5.4). Subdivision of the type 'press' will be necessary later, especially when deepdrawing operations, requiring multiple-working presses, have to be modelled. The machine is assumed to be simple: automatic feeding equipment or equipment for automated removal of produced parts is not supposed to be present.

From the point of view of cost estimation the following machine-related data are relevant: rate, operation mode, switch time, cycle time and machine use factors. The cycle time is the time for one stroke of the press, its determination will be discussed in 6.5.4. All machine use factors are 1.

The switch time, related to the action 'operate press', is not treated as a property of a specific press. In principle it is ignored because of its small influence, but the concept offers the opportunity to define a press independent switch time.

The operation mode can have one of two values: 'single stroke operation' or 'continuous operation'. Single stroke operation implies that the press is switched off and on after each stroke. Continuous operation is the opposite. The choice between the two operation modes depends on the cycle time and on the time to move strips between strokes. Continuous operation is possible only if the strip can be moved one step during the part of the cycle time in which it is free to move (not blocked by the tool).

Rates are defined for specific presses. It is considered that there may be several suitable presses available. This implies that a procedure to select a specific press from a list in the presses file is required to find the rate. Although this is essentially a process planning task, it has been integrated in the cost estimation module because the selection itself involves cost minimization. The selection procedure is described in paragraph 6.6.

6.2.4 Operators

One or more operators are always required. The operator level is defined to be 1 (rates on all levels are equal for the time being). The Human Attention Factor is set to 2 for the preparative actions: two operators are assumed to be necessary during tool handling. For the other actions the factor is 1.

6.2.5 Tool

A special made tool of the type 'progressive die' is always present. In principle, all tools that are commonly used in operations of the type described above belong to this type. Per definition this does not include tools for fine-stamping. In addition compound dies /Her-2, Gei-1, Rom-1, Wie-5/ or a 'compound step' as last step in a progressive die are not considered. Other special made dies
that have only one step can be conceived as progressive dies, as long as they are used in an operation where a strip or coil is moved through the tool. This implies that some facilities to guide and position strips have to be present. The precise definition of the tool type 'progressive die' needs further attention in future.

Apart from tool costs, that are considered in paragraph 6.7, four tool related data influence cost estimation: the type of positioning aid (TYPOS), the number of steps in the die (NSTEPS), the type of final operation in the die (FOTYP) and the tool multiplicity (M). The meaning of NSTEPS will be clear. The tool multiplicity is the number of complete parts produced per stroke.

As a part is produced by the die in different steps, positioning the strip before each stroke is of prime importance. Four different types of combinations of positioning aids (TYPOS) are distinguished:
1. Fixed Stop,
2. Fixed Stop in combination with Side Stops,
3. Side Cutter ('punch stop') at first step in die,
4. Two Side Cutters at first and last step in die.

Other combinations, including use of an 'american finger', are possible but not considered. The use of pilots for additional positioning accuracy is allowed for any combination. The TYPOS influences the time needed to move the strip and the effective use of the start and end of the strip /Wie-2/ as illustrated in figure 6.4.

Usually the part to be produced will be separated from the strip by the partial operation present in the last step of the die. This operation will be called the Final Operation. Five different types (FOTYP) are distinguished as illustrated in figure 6.5:
1. Cut Out is the same as 'blanking' and characterized by cutting along a closed contour. In contrast with the following FOTYP's it produces a Used Strip containing holes corresponding with the blank contours.
2. Normal Cut Off is characterized by cutting along one line that starts on one long strip edge and ends on the other.
3. Bridge Cut Off is characterized by cutting along two lines that each start and end on the long strip edges. The area between the two lines is the 'bridge' that represents scrap.
4. Two Part Cut Off is the same as bridge cut off, except that now the 'bridge' is also a valid output part, and not scrap. In contrast to the other FOTYP's it produces two parts per stroke per definition.
5. One Part Cut Off is the same as bridge cut off, except that now only the 'bridge' is a valid output part. Other material than the 'bridge' represents scrap.

From a time estimation point of view it is essential to note that cut out produces a Used Strip and that cut off does not. This fact directly influences the input/output time for the strip, and the determination of the OMF, which influences all times indirectly. Other FOTYP's than the five mentioned above can be dealt with by noticing whether there is a used strip or not.
Figure 6.4: Influence of the type of positioning aid (TYPOS) on the effective use of the start of the strip. The situations just before the first three strokes of the press are shown for a fixed stop and a side cutter.

Figure 6.5: Five types of final operations in a progressive die. Grey areas represent the cutting tool.
6.3 Strip LayOut

The definition of the locations and orientations of the blanks (the unfolded flat parts) in the strip is called the Strip LayOut. It can be conceived as a cutting plan that considers the outer contour of the parts only. Closely linked to this layout is the Tool LayOut that defines which features will be produced in each step of the die. Consequently tool layout also considers holes and bends. Some examples of strip layouts are presented in figure 6.6. For the cost estimation system the following layout-related parameters are relevant:
- the Pitch, defined as the distance over which the strip has to be moved after every stroke,
- the Object LayOut Length (OLL). This is the largest distance between any two points of the contours of blanks that are removed from the strip simultaneously, measured parallel to the long strip edges.
- the Object Layout Width (OLW). Similar to the OLL but now measured perpendicular to the long strip edges.
- the Overlap Length (LAP), defined as OLL - PITCH and valid only if this difference is positive (otherwise LAP=0).
- the Bridge Width between Objects (BWOO), defined as the smallest distance between any two blanks in the layout.
- the Bridge Width between Object and Strip Edge (BWOE), defined as the smallest distance between any blank and one of the long strip edges,
- the Strip Length (SL),
- the Strip Width (SW),
- the Strip Thickness (T),
- the Number of Rows (NOR). A row consists of all blanks whose orientations are the same and whose centres of gravity lie on one straight line parallel to the long strip edges.

For the time being BWOO and BWOE are both equal to T * BWOOF, where the latter variable is a material type dependent bridge width factor (typically with value 1.5). Strip LayOut definition is discussed in more detail in chapter 7.

6.4 Number of Strokes and Number of Parts per Strip

The determination of the Number of Strokes per Strip (NSS) and of the Number of Parts per Strip (OMF) has been discussed in detail in /Wie-2/. The resulting formula's give a good approximation, but they will not provide the exact number of strokes or parts in every possible situation. Only the main principle of reasoning will be explained here.

The strip can be conceived as consisting of a number of rectangles with a width equal to the strip width and with a length equal to the pitch. Every rectangle represents one possible stroke and M possible parts (M is the tool multiplicity). Whether all rectangles are actually used or not depends on the FOTYP and the TYPOS.

For FOTYP=1 (Cut Out) it is assumed that the strip end is always used, if the TYPOS permits it. For other FOTYPs (Cut Off) it is assumed that the strip end is never used because of the difficulties involved in moving this end under the tool (there is no used strip).
Figure 6.6: Definition of strip layout parameters for different types of strip layouts. Grey areas represent the cutting tool. See text for abbreviations and definitions.
The Number of Strokes per Strip is determined from:

\[ NSS = \text{INT}(SL/PITCH) + \text{CNSS} \]

The \text{INT} function indicates that fractions should be rounded downwards. \text{CNSS} is a correction that depends on both \text{FOTYP} and \text{TYPOS}. It is zero except for the following cases:
- \text{FOTYP} = 1 (Cut out) and \text{TYPOS} = 2 : \text{CNSS} = \text{NSTEPS} - 1
- \text{FOTYP} = 1 (Cut out) and \text{TYPOS} = 3 : \text{CNSS} = 1
- \text{FOTYP} = 1 (Cut out) and \text{TYPOS} = 4 : \text{CNSS} = \text{NSTEPS} - 1
- else (Cut off) and \text{TYPOS} = 1 : \text{CNSS} = -(\text{NSTEPS} - 1)

The Number of Parts per Strip depends on the NSS:

\[ \text{OMF} = (NSS - (\text{NSTEP} - 1) - \text{COMF}) \times M \]

In this formula, the term \((\text{NSTEP} - 1)\) reflects the number of strokes that is necessary before the first complete part is produced. During these strokes either incomplete parts are manufactured or there is no strip material under the final step in the tool yet (figure 6.6). \(M\) represents the number of parts per stroke. It is assumed that either none or all of the parts in a stroke are complete. \text{COMF} is a correction factor that depends on the overlap length. It is calculated as indicated below:
- no overlap (LAP=0) : \text{COMF} = 0
- \text{TYPOS} = 2,3,4 : \text{COMF} = \text{UP}(LAP/PITCH) \text{ (round fractions upwards)}
- \text{TYPOS} = 1 : \text{COMF} = \text{UP}(LAP/PITCH) - (\text{NSTEP} - 1) \text{ but minimal zero.}

6.5 Action Times

The estimation of action times is based on procedures and data supplied by Seemüller /See-1/, as interpreted by Haan and Hermans /Haa-2,Her-2/. The approach has been extended and adapted to make it suitable for use in computer programs and some minor details have been ignored /Wie-2/. The basic times presented here should not be used in any company without checking their validity.

6.5.1 Time for the main action

The time for one stroke of the press is called the Cycle Time (CT). The determination of this time is discussed in 6.5.4. The Main Time (MT) is directly related to the cycle time. As the MT needs to be defined per out-object we have to multiply the CT by the number of strokes per strip and then divide by the number of parts per strip:

\[ \text{MT} = \text{CT} \times \text{NSS} / \text{OMF} \]
6.5.2 Times for secondary actions

The Secondary Time (ST) consists of four components: the Input Output Time (IOT), the Movement Time (MVT), the Switch Time (SWT) and the Other Secondary Time (OST):

\[ \text{ST} = \text{IOT} + \text{MVT} + \text{SWT} + \text{OST} \]

In principle the last two times are neglected, but it is possible to define them as independent fixed constants.

The Input Output Time represents several secondary actions. It includes picking up the new strip, feeding it into the press, operating side stops, turning the strip and taking out the used strip and produced parts. The IOT is determined depending on Basic Input Output Times (BIOT's) as supplied by Seemüller for a Standard Strip with dimensions 1 * 30 * 1000 mm (and for materials with a specific mass from 7.0 to 8.9 kg/dm³):
- BIOT1 = 0.12 minutes for FOTYP= 1 (cut out: used strip),
or:
- BIOT2 = 0.10 minutes for other FOTYP (very small strip remainder).
and potentially:
- BIOT3 = 0.11 minutes if the strip has to be turned,
- BIOT4 = 0.01 minutes for every produced part to take out,
- BIOT5 = 0.02 minutes for every side stop to operate.

The influence of strip dimensions (weight) on these times is reflected by a Strip Dimension Factor (SDF). From the nomogram provided in /See-1/ a formula has been derived to determine this factor:

\[ \text{SDF} = 0.0088 \times \text{SL}^{0.44} \times (\text{SW} \times \text{T})^{0.5} \times \text{SWF} \quad \text{(minimal value is 1)} \]

SWF is the Strip Width Factor that depends directly on SW. It is defined for strip widths until 375 mm and then has a maximum value of 0.35.

For cut out as final operation the Input Output Time is determined from:

\[ \text{IOT} = (\text{BIOT1} \times \text{TURN} \times \text{BIOT3} \times \text{OUTFAC} \times \text{OMF} \times \text{BIOT4} \times \text{NSTOP} \times \text{BIOT5}) \times \text{SDF} / \text{OMF} \]

And for cut off as final operation (FOTYP 2 to 5):

\[ \text{IOT} = (\text{BIOT2} \times \text{OUTFAC} \times \text{OMF} \times \text{BIOT4} \times \text{NSTOP} \times \text{BIOT5}) \times \text{SDF} / \text{OMF} \]

TURN indicates whether Turn Cutting /Pic-2/ is applied (1) or not (0). It is only possible in combination with cut out. OUTFAC is the fraction of the produced parts that has to be taken out of the tool. It is zero unless FOTYP=4. In this case OUTFAC=0.50 is assumed. NSTOP is the number of side stops that has to be operated. It is zero unless TYPOS=2 and in this case its value is assumed to be NSTEPS - 1.
The Movement Time (MVT) represents the action 'move strip between strokes'. Its calculation is based on formulas for the Potential Movement Time (PMVT) that have been derived from nomograms in /See-1/:

Fixed Stop: PMVT = 1.34 * PITCH\(^{0.45}\) / 1000  \(\text{TYPOS: 1 or 2}\)

Side Cutter: PMVT = 0.92 * PITCH\(^{0.43}\) / 1000  \(\text{TYPOS: 3 or 4}\)

The PMVT is the actual time that is necessary to move the standard strip over a distance equal to the pitch. The influence of strip dimensions is taken into account by multiplying by the SDF. Judging from the description in /See-1/ it is probable that these times have been determined for cutting operations only. When the progressive die includes bending or drawing, it may become more difficult to move the Strip, especially when it has to be lifted to enable movement. This is neglected.

For single stroke operation the complete PMVT has to be taken into account. In continuous operation the strip is moved during the part of the cycle time in which the strip is not blocked by the tool. It is assumed that continuous operation is possible when:

\[
\text{PMVT} \times \text{SDF} \leq 0.75 \times \text{CT}
\]

In that case a part of the Potential Movement Time, defined by the Switch Factor (SF), is taken into account /See-1/:

Fixed Stop: SF = 2.44 * PITCH\(^{0.88}\) / 100  \(\text{maximum value 1}\)

Side Cutter: SF = 1.97 * PITCH\(^{0.61}\) / 100

For single stroke operation SF = 1.

The Movement Time per output part is now defined by:

\[
\text{MVT} = (\text{PMVT} \times \text{SDF} \times \text{SF} \times (\text{NSS}-1)) / \text{OMF}
\]

6.5.3 Time for preparative actions

No detailed basic time data have been found for the preparative actions. For the time being therefore, the time for these actions is fixed, independent of the weight or complexity of the tool and independent of the press. The value for the time has been taken from Ostwald /Ost-1/:

\[
\text{PT} = 0.8 \text{ hour} = 48 \text{ minutes.}
\]
6.54 Determination of the cycle time

CT determination is first discussed for mechanical presses. These presses are assumed to have a minimal number of strokes per minute (MINSPM) and a maximum number of strokes per minute (MAXSPM). These properties are stored in the presses datafile. If a press has a fixed SPM, both values are equal.

For a specific press the SM1-module will retrieve the data from file. If MINSPM and MAXSPM are equal, this value is selected and the cycle time is calculated from CT=1/SPM. The choice between single stroke operation and continuous operation will be made using this CT and the potential movement time.

If MINSPM and MAXSPM are not equal the module determines the highest SPM for which continuous operation is just possible (see 6.5.2):

\[ HSPM = 0.75 / PMVT \times SDF \]

Three situations are now possible:

1. HSPM is larger than MINSPM and smaller than MAXSPM. This implies that it can be selected and that CT = 1/HSPM using continuous operation.
2. HSPM is smaller than MINSPM. This means that the press can not move so slowly that continuous operation is possible. Consequently single stroke operation is selected, using medium speed: CT = 2/(MINSPM + MAXSPM).
3. HSPM is larger than MAXSPM. Continuous operation is possible even at maximum press speed: CT = 1/MAXSPM

Hydraulic presses have not been implemented due to difficulties in determining their cycle time. The main difference is that these presses do not have fixed SPM's. Instead, three speeds are often specified in the press documentation for:
- idle movement down ('free fall'),
- movement down while working,
- movement up.

With some ingenuity and insight the distances associated with these movements can be derived. Speeds and distances together permit the calculation of a cycle time. Such a model does not consider however that presses need time to accelerate and decelerate. The resulting cycle times turn out to be much too low.

A more realistic time/place diagram is shown in figure 6.7. Press documentation does not provide sufficient data however to calculate reasonable cycle times in this way. The answer can probably be found by combining the approach used above for mechanical presses with heuristic information on reasonable SPM’s for specific hydraulic presses. Such an approach has not been implemented.
6.6 Press selection procedure

It is assumed that several presses are available and that their properties are stored in the presses datafile. As rates are linked to specific presses, one press has to be selected. The selection process consists of two steps: testing presses for suitability and selection of the suitable press which leads to minimal operation costs.

Three suitability tests are performed (although more could be imagined):

1. Press Table Length (mm) \( \geq \) PITCH * NSTEPS,
2. Press Table Depth (mm) \( \geq \) SW + 2* (GuideWidth + FixWidth)
3. Press Capacity (kN) \( \geq \) T * (TENS/1000) * (CutLen + BendLen*0.2)

GuideWidth is the width in mm of the strip guides in the tool that define the strip position in the direction perpendicular to its main movement. FixWidth is the
width in mm needed to fix the die on the press table. Fixed values can be predefined for both widths.

TENS is the tensile strength of the strip material in N/mm. CutLen and BendLen are the actual total lengths to cut or bend. In future they should be derived from the tool layout. For the time being the BendLen is the sum of the lengths of all bendlines. The CutLen is approximated by adding:

- the part of the outer contour length that is actually cut,
- all lengths of hole contours and of open ended cut lines,
- the lengths cut by Side Cutters. For one punch (TYPOS=3) this is the sum of the pitch and the Width Allowance for Side Cutters. For TYPOS=4 twice this length is counted. The Allowance is calculated as $T^2BWOOF$, where $T$ the sheet thickness and $BWOOF$ the material type dependent bridge width factor.

The formula for capacity testing has been derived from the ones presented by Hermans /Her-2/.

It would be possible to select the suitable press with the lowest rate. In principle however it is possible that a more expensive press operates at higher speed and thus leads to lower operation costs. Therefore operation times are derived for each press and multiplied by the press rate to find operation costs. The press with the lowest costs is selected.

6.7 Tool cost estimation

Progressive dies could be estimated by treating them as products and using the detailed cost estimation system. This would require a complete tool specification however and that will usually not be present when the part to be produced by the tool has to be estimated. A more global method for progressive die cost estimation is needed.

From the point of view of DIDACOE the most interesting die cost estimation systems are those proposed by Nelson /Nel-1/, Harig (see /Mal-2/) and Prikos /Pri-1/. The method developed by Nordquist (see /Her-3/) requires a complete drawing of the die as input and is probably too detailed to be useful here. The former three methods are convenient because they try to estimate die cost using a part description and/or a strip layout as input.

Die costs are often estimated by considering the working openings in the cutting plate of the die, which correspond closely to the features in the part. A time related to each opening is derived, which not only represents the toolmaking hours required to produce this opening, but the hours to produce some or all of the related tool parts (the punch for instance) as well. Usually these hours are calculated for some standard situation. For special situations allowances are applied to the total time for all openings. Allowances reflect for example the influence of the type of tool material or of the dimensions of the die set. Material costs for the tool have to be estimated separately.
Two methods are commonly used to find the time per opening. Nelson uses tables in which times can be found depending on type of opening (type of part feature) and on some characteristic dimensions (hole diameter, length of bend). The features distinguished are: small, medium and large circular holes, L-, U- or V-shaped bends, straight, angular, curved or irregular blank shapes and extrusions. The ABC for die cost estimation presented by Prikos (figure 6.8) shows a similar classification.

Harig uses a more detailed way to describe opening contours, involving units. A unit is any uninterrupted straight, curved, or angular distance in the opening-contour not exceeding one inch in length (figure 6.9). For distances larger than one inch, each inch or portion thereof constitutes one unit. Thus a number of units can be derived for every opening. One unit corresponds with 5 hours of toolmaker time. Nordquist uses a similar approach for some parts of his die cost estimate.

By lack of time no progressive die cost estimation module has been implemented yet, which is a serious drawback. Before any implementation is attempted, the methods described above should be studied in more detail and an extensive literature study to find other methods should be performed. Probably some additional information would have to be acquired by contacting the referred authors. Moreover it should be noted that times in all methods are for traditional toolmaking. Electrical Discharge Machining (spark erosion) is not considered. New data that take into account this method should be gathered.

Considering required input for a die cost estimation module, it is to be expected that a tool layout and a classification of the contours of the working openings using one of the two methods above will be necessary.

Figure 6.8: Part of Prikos' ABC for die cost estimating. Toolmaker times associated with a part feature depend on type of feature and on its dimensions. (Source: /Pri-1/)
Figure 6.9: Definition of 'units' depending on the shape and dimensions of a working opening in the cutting plate of a die, as proposed by Harig. Every unit corresponds with five hours of toolmaking time. This time covers the production of all tool parts associated with the opening. (Source: Mal-2/)

6.8 Information Summary

A summary of all data involved in the specific module for production from strip in progressive dies can be found in appendix 2. The list presented below shows the required input data only. Default values, to be used by lack of other information, can be predefined for several of the input variables. This is indicated by (def) behind the variable. The use of the data is illustrated in figure 6.10.

Strip layout related data:
- strip length (SL) in mm (def),
- strip width (SW) in mm,
- strip thickness (T) in mm,
- pitch in mm,
- object layout length (OLL) in mm,
- number of rows (NOR) (as default value for tool multiplicity).

Tool related data:
- final operation type (FOTYP),
- whether turn cutting is applied or not (TURN, def),
- number of steps (NSTEPS, def),
- type of combination of positioning aids (TYPOS, def),
- tool multiplicity (M, default NOR),
- total cut length in mm (CutLen),
- total bend length in mm (BendLen),
- guide width in mm (def),
- fix width in mm (def),
- tool life in number of parts (def),
- interest time in years (def),
- repair and maintenance percentage (def),
- number of backup tools (def),
- basic tool cost (def). In future: tool layout, sufficient description of cutting plate working opening contours and other tool data to enable die cost estimate (not completely defined yet).

Assuming that material data can be retrieved from a materials datafile:
- material identification (necessary to access datafiles),
and in the file:
- tensile strength in N/mm², and
- bridge width factor (BWOOF).

Assuming automated press selection, the presses file should store per specific press:
- press rate in costs per hour,
- maximum capacity (force) in kN,
- minimal number of strokes per minute,
- maximal number of strokes per minute,
- press table length in mm,
- press table depth in mm,
otherwise user specifies rate only.

Basic time data that must be present in times datafile:
- BIOT1 in minutes per strip for strip input/output in Cut Out (now 0.12),
- BIOT2 in minutes per strip for strip input/output in Cut Off (now 0.10),
- BIOT3 in minutes per strip for turning during Turn Cutting (now 0.11),
- BIOT4 in minutes per part for take out of produced parts (now 0.01),
- BIOT5 in minutes per action for operating side stops (now 0.02).

Other variables whose values have to be specified in files:
- Switch Time (SWT) in minutes per 1000 switches (now 0.0),
- Other Secondary Time (OST) in minutes per part (now 0.0),
- Preparation Time (PT) in minutes per batch (now 48.0),
- Rejection Factor (now 0.02),
- Operator level (now 1),
- Machine Use Factors per type of action (now all 1),
- Human Attention Factors per type of action (now 2 for preparation and 1 for secondary and main).
Figure 6.10: Information summary for time estimation module
7 Basic Feature Description and Recognition

7.1 Introduction

Programs for automated process planning and programs for the generation of manufacturability information need a high-level part description to be able to reason about the production of a part. As indicated in paragraph 4.5 the elements of such a description will be called features. A set of features to describe a class of sheet metal parts has been developed. The resulting description can be used for parts that contain the features indicated in paragraph 4.8. It consists of two parts: the 'basic feature description' and the 'advanced feature description'. The former will be explained in this chapter.

The basic features provide a method of description for completed sheet metal parts and for less worked objects (unfinished sheet metal parts). They are the base for the derivation of advanced features. Different advanced feature descriptions for different purposes can be derived from the basic description. The next chapter will consider the use of advanced features for automated strip layout generation. A full list of feature-related data can be found in appendix 2.

7.2 Feature descriptions

7.2.1 Compound basic features

Four compound basic features are distinguished (figure 7.1):
- the Blank is the highest level feature. It represents the shape of the whole part and is used to store properties that are related to the object as a whole, such as sheet thickness, total effective area, total cut length and total bend length.
- the Outer Contour describes the perimeter of the unfolded sheet metal part. Any blank always and only has one Outer Contour (OC).
- Holes represent areas inside the OC where material is absent: it has been cut out in some way. Per definition Hole Contours do not intersect or touch the Outer Contour. Zero, one or more holes can be present in the description for a part.
- a Face represents an area within the blank that is bounded by one closed Face Contour and that lies in one flat plane in the folded sheet metal part. Zero, two or more faces can be present. The definition of faces will be illustrated in more detail in 7.2.3.

The Outer Contour, Hole Contours and Face Contours are described by a sequential list of elementary basic features, which can be conceived as line segments. Starting from any point, going either clockwise or anti-clockwise, the list describes a closed contour. The property 'Side' for any contour defines the
Figure 7.1: Compound basic features for the description of a class of sheet metal parts. The Blank is the highest level feature which represents the complete part. It is conceived to consist of one Outer Contour, zero, one or more Holes and zero, two or more Faces. Each contour is described by a sequential list of elementary basic features.

direction of description. A value +1 indicates clockwise description, -1 indicates anti-clockwise description. Side +1 implies that, walking along the contour in the direction of description, the feature (the material or the hole) is always on the right-hand side.

7.2.2 Elementary basic features

Five different types of elementary basic features are distinguished. They can be conceived as line segments or as contour elements:
1 Straight cut line segments (LIN),
2 Curved cut line segments with constant radius (RAD),
3 Complete cut circle (CIR),
4 Curved cut line segments of second degree with non-constant radius (CUR),
5 Straight bend line segments (LIB).
Figure 7.2: Elementary basic features and their description. Four types of cutline-segments are distinguished: straight segments (LIN), parts of circles (RAD), complete circles (CIR) and other second degree curves (CUR). Most segments have a startpoint (SP) and an endpoint (EP), which implies that they have a direction. For RADs this direction is defined explicitly (clo for clockwise and ant for anti-clockwise). RADs and CIRs have a centrepoint (CP) and a radius (R). CURs have a helppoint (HP) which is the intersection of the tangents to the curve in start- and endpoint. Associated with this HP is the weight which determines the shape of the curve (the flatness), see /Fau-1/. Direction dependent angles (A) are defined for LINs as shown. For RADs and CURs two angles are defined in the same way: one for the tangent in SP (AS) and one for the tangent in EP (AE). One type of bendline-segment is distinguished: the LIB. It is completely similar to a LIN but has two additional properties: the BendAngle and the BendRadius.
The Outer Contour and Hole Contours are described by a sequential list of LINs, RADs and/or CURs or by one CIR. Face Contours are described by a sequential list of LINs, RADs and/or CURs and at least one LIB.

The types of contour elements are described as indicated in figure 7.2. Note that all elements have a direction, from a startpoint to an endpoint. Depending on the type of element additional properties are defined to complete its description, like helppoint and weight for a CUR or centrepoint and radius for a RAD. Direction dependent orientation angles are defined for the segments as shown.

7.2.3 Bends, bend lines and faces

Ideally bending can be conceived as the rotation of one face with respect to another face around a line (the bend line) over a specified angle (the BendAngle). Figure 7.3a shows a top view of the unfolded part and side-views of the folded parts depending on the bend angle. A positive angle indicates that

![Diagram of bend line and bend zone with equations for calculation]

(a) Idealization  
(b) More realistic

Figure 7.3: An idealized (a) and a more realistic (b) model for bending. The BendLine is the center of the BendZone. The BendAngle is the angle over which one of the faces has to be 'rotated', not the angle between the faces. The angle is positive if movement is 'into the paper'. The BendRadius is the inside radius (R1). Faces include the corresponding half of the bendzone.
one face will move 'into the paper' when the plane of the other face corresponds with the plane of the paper. For the final geometry of the part it does not make any difference which face is supposed to move. The bend angle is the angle over which one of the faces has to be 'rotated', not the angle between the faces.

A more practical model of bending, although still somewhat idealized, takes into account that the faces have a thickness larger than zero and that their material allows limited deformations only. Consequently the change in angle is not applied at once: the angle 'grows' in a transition zone (the BendZone) from zero at the end of the remaining face to the desired angle at the start of the moving face. Associated with this more practical situation is a BendRadius, which is per definition the inside radius (Ri) as illustrated in figure 7.3b. The bend line now represents the centre of the BendZone. The two halves of this zone on both sides of the bend line have equal widths and are conceived to be part of the corresponding faces. The widths can be calculated using the so called neutral radius (/Her-2,Gei-1/). In the feature description, bend angle and bend radius are properties of bend line segments (LIBs).

If a part does not contain any bend lines, no faces are defined. Otherwise the number of faces will usually be one larger than the number of bend lines, but there are exceptions. The selected examples presented in figure 7.4 clarify the definition of faces. Note the following points:

- part A contains only one bend line and two faces. As the bend line is interrupted by a hole, it consists of two segments. Note that both LIBs should have the same angle and radius.

- parts B, C and D contain two bend lines and three faces. The bend lines now each consist of one LIB. Note that the two LIBs in each of the parts can now have different angles and radii. They need not lie on one line.

- in part C the cut line that separates the two small faces is extended to the end of the bend zones. From the point of view of definition of face contours the extension (on the right-hand side of the bend line) is neglected.

- the decomposition of part E shows that a face is always defined as the smallest possible area that is completely surrounded by a closed contour consisting of bend line and cut line segments: face 1 is not supposed to lie on face 2. The same part of the Outer Contour or of a Hole Contour can not belong to more than one Face Contour. In contrast, face 1 is supposed to lie on face 2 for part F. Here, the contours of both faces do not have any part in common.

- part G is essentially the same as part A, but the hole has a different shape. This shows that a face can extend 'to both sides' of the bend line in very special cases.

- part H poses some problems. Normal geometry requires that only two bend lines are distinguished. The middle of the part will typically be conceived as a unit, although it consists of two separated halves. It can not be described as a face with one closed face contour however. Therefore two faces are distinguished in the middle part for the time being.
Figure 7.4: Examples of the decomposition of blanks into outer contour (OC), holes (H) and faces (F), to illustrate the definition of faces and face contours. See the text for remarks.
Bends are distinguished from faces. A bend is a combination of two or more faces. Different types of bends can be defined depending on the relations between the faces: U-, V-, L-, Z-shaped bends. Faces can be defined independent of manufacturing. For bends this is more difficult. During manufacturing, three faces that together look like a 'U' can be produced at once like a 'U' or in two separate steps like two 'L'. So, faces are basic features, while bends are advanced features.

7.2.4 Special conditions and tolerances

Two types of special conditions and tolerances can be distinguished. The first group includes all those that can be treated as a property of one of the basic features. Examples are:
- heat treatment condition, surface condition, flatness tolerance or sheet thickness tolerance for the whole blank,
- edge conditions for the Outer Contour, Hole Contours or for any cut line segment,
- dimensional tolerances on the radius for a CIR or RAD,
- tolerances on the bend angle or bend radius for a LIB.

The second group includes all those conditions and tolerances that are associated with a relation between two or more basic features. They can not be stored as a property of a basic feature but have to be treated as separate features. Examples are:
- a parallelism tolerance between two LINs,
- a positional tolerance for the centrepoint of a circular hole with respect to the centrepoint of another circular hole.

Although some space was reserved to store special conditions and tolerances in the basic features (see appendix 2), the topic was not given sufficient attention yet. For the time being the DIDACOE-implementation ignores tolerances and special conditions. Of course, this is a serious drawback, as they have a major influence on process planning decisions and on manufacturing costs.

7.3 Feature Recognition Program

7.3.1 Input: SMD and the corrected development

The current DIDACOE-implementation uses the CAD-system Medusa and its Sheet Metal Design module (SMD, /Cam-1/). As shown in figure 7.5, this module consists of three programs: an Unfolder, a Bend Allowance program and a Folder. The designer specifies the Ideal Development directly using 2D Medusa or he creates a 3D model which is processed by the Unfolder with the same Ideal Development as result. This development can be edited if necessary. The Bend Allowance program adjusts the widths of the bend zones (and all associated dimensions) in the ideal development considering a definition of the neutral radius supplied by the user. It produces a corrected development which
Figure 7.5: Main programs in MEDUSA's Sheet Metal Design System. The corrected development produced by the Bend Allowance program is the input for the Feature Recognition Program (Source: /Cam-1/, adapted)

Texttype TBS →

THICK 2
ANG 90.0
RI 5.0
MAT PEPOO

Texttype TBG →

ANG -45
RE 10

Cutlines: linetype LPO
Bendlines: linetype LP1

Figure 7.6: The most important drawing elements present in a description of the corrected development. Texts are necessary to define sheet thickness, bend radius and bend angle in the 2D representation. Texts of type TBS define default values. Texts of type TBG, with their reference point on a bendline, define local values for radius and angle that override the default.
can be edited again if necessary. The Folder can create the final 3D model from this corrected development.

As explained at the end of paragraph 4.5 the Feature Recognition Program takes the 2D corrected development as input. In general this drawing will contain the following information (figure 7.6):

- lines of type LPO that represent cut lines. These lines define the Outer Contour, Hole Contours and 'open' cut lines that start and/or end on the contours, on other open cut lines or on bend lines,
- lines of type LP1 that represent bend lines: the centres of bend zones,
- one text of type TIS that represents a registration code or number,
- several texts of type TBS that define default values valid for the whole part, unless specified otherwise locally. These include:
  - 'THI' or 'THICK' followed by a value for sheet thickness,
  - 'RI' followed by a value for inner bend radius,
  - 'RE' followed by a value for outer bend radius,
  - 'ANG', 'ANGLE' or 'BEND' followed by a value for bend angle,
- several texts of type TBG that define local values to replace the defaults specified by the TBS-texts. Values for bend radius and bend angle are defined this way. The texts have their reference point on the bend line,
- other texts and lines that are ignored or that will not be discussed here, see the SMD User Manual /Cam-1/.

In addition to texts that have a meaning in SMD, other TBS-texts can be put in the drawing to specify type of material and production quantities.

From the point of view of recognition of basic features this input includes some difficulties:

- although the Outer Contour and Hole Contours may look closed on the screen, a study of the underlying data may reveal that the contours consist of a collection of open ended lines. These lines are not related to each other in the datastructure,
- a Medusa line consists of points and line segments between those points. It is possible that a line contains segments with length zero (two coinciding points). It is not desirable to include these in the feature model,
- bend lines and open ended cut lines start and/or end on other lines. In the former lines these start and end points are clearly defined, but in the latter they are usually not. That is: a bend line can end on the middle of a contour segment. This implies that existing Medusa segments sometimes have to be split (introduce new points in a line) to define the segments required for the description of Face Contours,
- on the screen it may be evident that one line ends on another line, while the underlying data show that there is in fact a small distance between the two. On several occasions these and similar accuracy problems have to be solved,
- from the point of view of sheet metal manufacture it is of great importance to note whether a curved line segment is part of a circle (RAD) or not (CUR). Although Medusa uses two different representations for curved segments, one typical for RADs and the other typical for CURs, both representations can be used for both types of segments.

101
- Medusa uses as much as four different representations for circles. This confuses even the routines of the CAD-system itself.

Note that most of these 'problems' are typical for 2D CAD representations. On normal technical drawings on paper, interpreted by human beings, they would not cause problems. Probably most of the difficulties would also not be present in 3D solid models, as these require more consistency and accuracy.

7.3.2 Output: NXP—database—format files

The Feature Recognition Program produces the basic feature description as defined above. For ease of implementation this description is not stored in a real database using a database management system, but in a collection of ASCII-files. These files use the NXP-database-format (figure 7.7) that can easily be read by other programs that have to access the data. The NXP-files only contain the data for the last part that was analysed by the Feature Recognition Program. Old data in the files are deleted as soon as the next analysis provides new data. A total of 19 files is used to store the description. Five general files define the Main Object, the detailed estimated MakePart, the material type and properties and the shape type. The shape type is now always 1, which indicates the basic feature description introduced above. Fourteen sheet metal specific files exist. There is a separate file for each type of feature: Blank, Outer Contour, Hole, Face, Lin, Rad, Cur, Cir, Lib and Point. Four additional files store the feature relations: the sequential lists of contour elements and the features on faces (not discussed here). In addition to the NXP-files the feature recognition program produces a revised drawing next to the original one, that remains unchanged. In this file zero length segments have been removed, lines have been combined to real closed contours if this was necessary, and points have been added to lines where other lines end on them. In addition lines associated with specific features are now grouped on specific layers in the drawing (for instance: all hole contours on layer 67).

7.3.3 Program outlines

The Feature Recognition Program has been implemented in Fortran 77 on a VAX under VMS. The program uses the Medusa Data Access Routines (DARS, /Cam-2/) to access the data in the file produced by the SMD-module. These routines perform the following tasks for example:
- open or save a file,
- search for a next element in the sheet that meets specified conditions,
- get data associated with a current line or text,
- create new line or text and define its data,
- delete current element,
- check whether one line is inside another line or not,
- check whether a point is inside, outside or on a closed line,
- get area enclosed by a line.
Figure 7.7: Illustration of the NXP-database-format to store the basic feature description supplied by the Feature Recognition Program. Every file resembles a 'normal' database table with fields. Not all actually present fields are shown. The feature type (FT) in the OCE-file points to a file (1 for LIN, 2 for RAD). The feature number (FN) points to a specific entry in this file. In the same way pointnumbers SP (start), CP (centre) and EP (end) point to specific entries in the POINT-file. Other abbreviations: PM is Perimeter Type, SD is Side, EC is Edge Condition, ANG, SAN and EAN are angles, DR is the RAD-direction, RAD the radius, RT a pointer to a radius tolerance (not handled yet and thus unknown) and PT a point-type. OC, LN, RD and PN are sequential reference numbers in the files.
Figure 7.8: Main steps in the Feature Recognition Program. The program has been implemented in Fortran 77 on a VAX under VMS and uses the Medusa DARS-routines to access data in the Medusa-file that contains the part description.
The main steps in the Feature Recognition Program are shown in figure 7.8. A more extensive description can be found in /Wie-9/. Complete information is contained in comments in the program source text.

7.4 Evaluation

Considering the current implementation of the Feature Recognition Program four remarks can be made:

1) The DARS-routines offer less advantages than expected. It is difficult to store the positions of specified elements in the hierarchy of the drawing for later use. As soon as the drawing is changed, the positions of all elements can change. In addition DARS-routines do not always provide the information in the required way. There is a routine for example that checks whether a point lies on a line, but it does not tell on which segment in the line it is. This implied that a lot of additional routines had to be developed.

2) Considering the first point, one of the reasons to choose for a Medusa-specific Recognition Program (4.5) becomes doubtful. Maybe the work involved in using an IGES-file as input for the program would almost have been the same as the work now done to write a program using DARS-routines.

3) The program has been written in such a way that the facilities offered by DARS-routines are exploited as much as possible (area calculation for example). The consequence is that the use of DARS-routines is scattered all through the program. It is probably better to use DARS-routines once, at the start of the program, to extract all relevant data from the Medusa-file, and to resolve all other tasks by own developed routines. In this way only a very small part of the program would need to be adapted for use in combination with other CAD-systems.

4) The basic feature description and its storage in the NXP-files are simple and straight forward. It should not be too difficult to generate the same output starting from other CAD-systems. The basic feature description can easily be extended to include features like louvers, tabs, extruded holes and the like. It should also be possible to include special conditions and tolerances. In general the basic feature description forms a suitable base for the derivation of all kinds of advanced features.
8 Strip Layout generation and Advanced Features

8.1 Introduction

The specific cost estimation module for the production of sheet metal parts from strip in progressive dies requires strip layout and tool layout information as input (paragraph 6.8). The strip layout specifies strip dimensions and the positions and orientations of the unfolded (developed) sheet metal blanks in the strip. It fixes variables like strip width, pitch, object layout length, number of rows, bridge widths and the like. The tool layout describes the partial operations that are to be performed in each step of the die. It fixes variables like number of steps in the die, type of positioning aid and shape and position of cutting plate working openings.

Although a distinction is made between strip layout and tool layout, the two are very closely related. The choice of the type of final operation for instance, or the determination of the number of rows, immediately influence both layouts. Although strip layout precedes tool layout, the former can not be made without considering some important aspects of the latter.

Layout design is a detailed process planning task that requires expert knowledge and experience. Within the context of DIDACOE, these tasks can not be left to the designer. Therefore a completely automated Strip Layout Generation program should be developed. A prototype for such a program has been implemented and will be described here. Tool layout has not been handled yet, but some remarks will be made.

8.2 Existing programs for layout design

As noted by Branson /Bra-2/ two approaches to computerized die design (which includes strip layout definition) can be distinguished: 'screen assisted drawing' and 'fully computerized drawing'. Programs using the latter approach design a layout automatically, given a suitable description of the developed sheet metal part. The first type of program offers facilities that enable the user to design a strip or tool layout interactively in a graphical environment. Typically, such programs permit duplication, rotation, mirroring and translation of blanks. They can check user designed layouts for violation of minimal bridge widths and supply material use information. Some systems also offer facilities to define and position tool elements like punches.

The interactive type of program is most popular. These systems are relatively easy to implement, speed up layout design considerably and permit the user to control the whole process. Typical examples include Siemens' WEKO-program /For-1/, the Euclid-linked COSAC-program made by CRITT /Cri-1/, the programs described by Nakahara /Nak-1/, Fogg /Fog-1/ and Cornely /Cor-1/ and probably the TDC-program briefly described by Da-Xin /Dax-1/.
The programs described by Blume /Blu-1/ and Shirai /Shi-1/ seem to be partly automatic, while Blank's OLETIS program /Bla-1/ and the PDDC-program referred to by Branson /Bra-1/ are completely automated. Especially the PDDC-program description looks promising, but detailed information was not available.

From the point of view of strip layout design in DIDACOE a number of remarks can be made regarding the referred programs:
- as far as can be judged from the descriptions, all programs aim at material use optimization, which will typically lead to one best solution. Effects of strip layout on operation costs and tool costs seem to be neglected.
- all automatic or semi-automatic programs seem to consider strip layouts intended for cut out as final operation only. Yet, a complete automated layout design program should include cut off situations as these are the most interesting from a cost minimization point of view.
- just like most process planning programs, the above approaches are aimed at providing information for manufacture. DIDACOE primarily intends to supply information for the designer. This implies that the layout program should provide recommendations for changes in the design next to merely accepting the design and producing a layout. To enable this, the program should have a thorough understanding of both the features in the part to be produced and the factors that influence strip layout design. This knowledge is probably not present in any current program.
- considering the objective of DIDACOE, interactive layout design programs are not acceptable as they expect too much knowledge from the part designer. Some of the programs have clearly been developed for tool designers.

8.3 Outlines for the new strip layout generation program

8.3.1 General program philosophy

Considering the above it was decided to develop a new program from scratch. The following points describe the underlying philosophy:
- considering the main DIDACOE objectives and assumptions the program can not prompt the part designer for input. The basic feature description and the additional material and quantity information provided by the feature recognition program are the only input. Starting from this input the program generates layouts completely automatically.
- strip layout does not only influence material costs, but tool costs and operation costs as well. The DIDACOE objective should be to minimize overall costs. This could be achieved by estimating costs for all acceptable layouts. One problem is to recognize which layouts are acceptable (technically possible, leading to the desired output). An other problem is that there will usually be many acceptable solutions, requiring a lot of estimates. The objective of a layout design program should be to select the most promising acceptable layouts. In most cases this will imply that the program generates a limited number of layouts (but more than one), leaving it to the cost estimation module to choose the cheapest solution.
- the most promising blank orientations (one or a few) can often be derived from the outer contour shape and from some other part characteristics. Many existing programs neglect this. Usually a brute force angle variation technique is applied to find the layout for cut out with maximum material utilization. The general philosophy of the new program is to use this brute force technique as a default solution only. Prior to using this default, the program will examine the outer contour shape of the blank (and other part characteristics) in an attempt to derive strip layouts by 'reasoning'. In particular it will explore possibilities for cut off. Other than the brute force method this approach gives the program sufficient information to comment on the part design from a layout point of view (manufacturability information).

- given the limited time and the complexity of the problem it will only be possible to develop a prototype layout design program with limited 'reasoning capabilities'. The future objective will be to extend the number of cases that the program can handle by reasoning and to limit use of the default angle variation technique as much as possible.

- considering the desire to gain practical experience in working with knowledge-based system techniques, the complexity of the relations between all factors that influence strip layout design decisions and the need for a program that can easily be adapted and extended, it was decided to implement the new program in Nexpert Object: an environment for the development of knowledge-based systems (see chapter 11).

8.3.2 The main output: Strip Layout description

The correct development of the sheet metal part (the blank) is described in an Object Coordinate System (O). The strip will be described in a Strip Coordinate System (S). Both systems use normal cartesian coordinates. The blank can be anywhere in the object system, but the position of the strip in its system is fixed as shown in figure 8.1a. The strip system origin coincides with one of the corners of the strip. The short edge at the start of the strip is described by X=0 and positive Y-values between 0 and the strip width (SW). Both long strip edges have positive X-values between 0 and the strip length (SL).

The location of the first blank in the first row of the layout is defined by (figure 8.1b):

- the orientation angle for the first row, ORANG(1), which is defined as the angle over which the positive XS-axis has to be rotated to make it coincide with the positive XO-axis (assuming that origins of both systems coincide). The angle is positive for anti-clockwise rotation and defined between minus \( \pi \) (exclusive) and plus \( \pi \) (inclusive),

- the coordinates (XS0, YS0) of the object system origin in the strip coordinate system.

A point P with coordinates (XOP, YOP) in the object system now has coordinates (XSP, YSP) in the strip system, defined by:

\[
\text{XSP} = \text{XOP} \times \cos(\text{ORANG}(1)) - \text{YOP} \times \sin(\text{ORANG}(1)) + \text{XS0} \\
\text{YSP} = \text{XOP} \times \sin(\text{ORANG}(1)) + \text{YOP} \times \cos(\text{ORANG}(1)) + \text{YS0}
\]
Figure 8.1: Strip Layout Definition. (a) Object coordinate system and strip coordinate system, (b) location of first blank in the first row of the layout, (c) location of the N-th blank in the first row, (d) location of the first blank in the M-th row.
Other blanks in the first row have the same orientation angle as the first blank. In addition the origins of the object systems of all blanks in the row lie on one straight line parallel to the long strip edges. The distance between any two adjacent origins is equal to the pitch (figure 8.1c). This implies that the whole first row in the strip is described by ORANG(1), XSO, YSO and PITCH.

The location of the first object in the M-th row of the layout is described by (figure 8.1d):
- the orientation angle ORANG(M) for this row,
- the location of its object system origin relative to that for the first blank in the first row:
  \[ DXS = XSO(M) - XSO(1) \text{ and } DYS = YSO(M) - YSO(1) \]
- the variable MIR with value -1 if the blank has been mirrored and value +1 if it has not been mirrored. The object system X-axis is the line for mirroring.

The coordinates (XSP, YSP) of a point P in a blank in the N-th position of the M-th row of the layout can now be derived from its object system coordinates (XOP, YOP) by using:

\[
\begin{align*}
XSP &= XOP \times \cos(\text{ORANG}(M)) - MIR \times YOP \times \sin(\text{ORANG}(M)) + XSO + DXS(M) + (N-1) \times \text{PITCH} \\
YSP &= XOP \times \sin(\text{ORANG}(M)) + MIR \times YOP \times \cos(\text{ORANG}(M)) + YSO + DYS(M)
\end{align*}
\]

This defines the complete layout except for minimal strip width (MSW) and strip length (SL). The latter is supposed to be a predefined program constant. MSW has to be generated as output. It is the minimal required strip width, considering final operation type and bridge width. It does not include width allowances that might be necessary for side trimming stops. At the moment availability of strip widths is also not taken into account.

The above implies that the output to generate consists of minimal strip width, number of rows, pitch, and XSO and YSO for the first blank in the first row. Additionally for every row: orientation angle, DXS and DYS for the first blank in the row and whether blanks in the row are mirrored or not.

The current implementation stores some additional information for a strip layout. The final operation type, although theoretically a tool layout property, is so important during strip layout that it is more convenient to conceive it as a strip layout property. Overlap length and object layout length, required by the cost estimation module, are derived from the above layout data and the outer contour description. Temporary values for material use percentage and actual total cut length have been added to this list. The former is recalculated later in the cost estimation module using the actual number of parts produced per strip. The total cut length is temporarily used during press selection, by lack of tool layout data.

All data are stored in two NXp-database-format files, one for data per layout and one for data per row (see appendix 2).
8.3.3 Additional output: Manufacturability Information

Manufacturability information has not been studied in detail yet, but it is probable that its generation will consist of activating messages from a predefined list of messages. They can be textual, graphical or both. Some messages will have a general nature, meaning that they are not clearly related to features in the part. In most cases however a message will be associated with a specific feature or with the relation between two or more features. In these cases the messages will have to be presented in the part drawing, linked in some way to the associated features, similar to the presentation used by Baumann (see figure 3.8). Considering this, a unit manufacturability information will probably consist of:
- a reference number for a textual message,
- a reference number for a graphical illustration,
- one or more pairs of x,y-coordinates that indicate the feature(s) in the part drawing to which the message has to be linked.

8.4 Influence of Outer Contour shape on layout design

The shape of the outer contour is often the most important factor that influences strip layout design. Therefore it is treated here first, separated from other influence factors. The descriptive explanation in this paragraph (and in the paragraphs 8.5 and 8.6) is intended as an introduction. Some of the concepts will be defined better in subsequent paragraphs.

8.4.1 General principles for cut off

The different cut off variants have been described in paragraph 6.2.5 and in figure 6.6. Considering the definitions presented there, cut off is possible only if the outer contour shape meets some conditions. For all variants the contour should have two parallel edges that can coincide with the long strip edges. Of course, the whole outer contour should lie on or between the lines defined by these edges.

Assuming that two suitable parallel edges are present, it is always possible to define two other edges that connect their ends. In the layout these connecting edges will start and end on the long strip edges, implying that they represent the cut off lines.

Normal cut off requires that the two connecting edges are identical, because the same punch cuts both edges. Two Part cut off is possible more often because two different cut off lines are allowed. Three solutions are possible:
- all blanks belong to the same row in the layout,
- blanks in the second row have been mirrored,
- blanks in the second row have been rotated over 180 degrees.

In all cases the connecting edges have to match each other in some way as illustrated in figure 8.2.

Bridge cut off and One Part cut off do not require any particular relation
Figure 8.2: Examples of layouts for cut-off: (a) Normal cut-off, (b), (c) and (d) Two Part cut-off, (e) and (f) Bridge cut-off. The part to be produced has two parallel edges and two connecting edges. In the first four cases the latter edges are identical in some way: they match either directly or after mirroring or rotation. The last layout is not advantageous due to the large bridge (=scrap).
between the connecting edges. The two cut off lines each define one edge only. Note however that the bridge (scrap) can become very large if the connecting edges have inconvenient shapes. Therefore cut off is not always useful in these situations.

These stringent conditions would lead to a very limited number of cases in which cut off can be applied. Fortunately the restrictions can be relaxed while still using the same main principle (figure 8.3).
The first relaxation is to allow the parallel edges to be interrupted by notches. The notches can easily be made in a step in the die that precedes the final step. Consequently they do not interfere with the cut off principle.
The second relaxation is to allow connecting edges to be not completely identical. In this case parts of the edges 'match', while the non-matching parts describe areas in the strip that can be punched out in steps previous to the final one. The non-matching parts can occur both in the middle and at the ends of the connecting edges.
Considering sheet metal production practice these relaxations are necessary. When the interruptions or the non-matching parts become very large however, cut off may not be advantageous any more.

84.2 Special cut off situations

Outer contours that contain (interrupted) parallel edges and (partially) matching connecting edges potentially are the most interesting shapes for the generation of layouts for cut off. They are not the only shapes that can lead to advantageous cut off however.

In particular, there are a number of specific shapes for which convenient cut off layouts are well known. Examples for L- and T-shaped contours are shown in figure 8.4/a/b. Similar advantageous layouts for other contours can be found in literature or they can easily be imagined. The common characteristic of these contours is that they are suitable for nesting in a strip: parts of the contours match other parts in such a way that a very good material utilization can be obtained. In contrast with the general approach above, these contours can be conceived as specific cases which have specific associated layouts. Similar to the relaxations described above, small distortions of these ideal shapes can be allowed while still using the same standard layout solution.

More in general, as an extension to the principle described in the previous paragraph, the condition that a part should have two parallel edges that can coincide with the strip edges can be relaxed. Cut off layouts are possible even if there is only one such edge or if there are none at all (figure 8.4c,d). Some of the resulting layouts may be advantageous, but using this relaxation the difference between cut out and cut off may become very small, both from a material use point of view and from a tool cost point of view. This is especially true when combining the relaxation with Bridge cut off or with One Part cut off.
Figure 8.3: Examples of layouts for cut-off. Notches and/or holes are punched out in preceding steps in the die. Compared to the parts in figure 8.2, parallel edges are now interrupted and the 'match' between connecting edges is not complete. Non-matching parts describe holes between the blanks.
Figure 8.4: Special examples of layouts for cut-off. In these cases the part does not contain two parallel edges, or they are not used to coincide with the long strip edges as was done in the previous two figures.
8.4.3 Shapes for cut out

In principle, cut out does not require any particular outer contour shape (except for the corner-angles discussed below): any shape can be made by cut out. This implies that layouts for cut out are the default solution, to be used when cut off is not advantageous. Consequently, the shape of the outer contour does not have to be examined to check whether cut out is possible.

The main problem during the generation of layouts for cut out is to determine the best orientation angle and the associated pitch. Usually the brute force angle variation technique is applied to find the solution with optimum material use. Use of this technique can be avoided by taking into account the main shape of the outer contour. In a lot of cases it is possible to derive one or a limited number of promising orientation angles from the outer contour shape. A number of specific perimeter types, with associated layouts, can be defined for this purpose.

8.5 Influence of bend lines on layout design

8.5.1 Bend lines and grain direction

Strip material will usually have been produced by cold rolling. Consequently material properties are direction dependent. In general it is recommended to make bend lines lie perpendicular to the grain direction (the direction of rolling). If this is not possible, for example because the part contains several bend lines that are perpendicular to each other, the angle between bend lines and grain direction should preferably be at least 45 degrees /Bra-1, Bra-1/. Grain direction usually runs lengthwise in the strip.

These general rules can be detailed by taking into account material properties and bend radius. The minimal angle required between bend line and grain direction typically depends on material tensile strength and brittleness and on the bendradius/thickness ratio /Bla-1, Her-2/. Figure 8.5 illustrates this. Grain direction can also be important in relation to the strength of flat parts. If necessary, the drawing should specify constraints /Bra-1/. This fact will not be considered here however.
Indications for the orientation angle derived from grain direction considerations can be formulated positive or negative. The first approach will lead to the definition of one or two fixed angles. The negative formulation will result in a range of 'forbidden' angles, which offers more flexibility.

8.5.2 Special layouts for bending

As explained before, the basic feature description for sheet metal parts may contain faces. Before these faces can actually be bent they should be free to move. This implies that the material on the outside of a face contour should be removed prior to the forming operation, except of course, along bend lines.
In a progressive die bending operations are usually performed before the blank is completely separated from the strip, although this is not strictly necessary. The material on the outside of faces that have to be bent can not be used to keep the blank attached to the strip, as this material will be punched out to free the faces. Consequently there should be at least one face that remains in the plane of the strip: the reference face. Only strip material that lies outside this face, along cut lines in this face contour, can be used to provide the bridge between blank and strip. The final operation removes this bridge after all other partial operations have been performed, to separate the part from the strip.

This principle can make cut off solutions impossible. In layouts for cut out it may require larger bridge widths between blanks and between blanks and strip edges. This influence of the presence of bend lines is illustrated in figure 8.6, while figures 8.3, 6.4 and 8.7 provide practical examples of strip layouts for parts including bends.
Figure 8.6: Influence of the presence of bendlines on layout design: (a) normal cut-off layout for part without bends, (b) and (c), possible layouts for the same part containing bends. Bending is usually performed while the blank is still attached to the strip. Faces that have to be bent should be cut free prior to the forming operation (b) or combined cut/bend-tools should be used (c).

8.6 Other influences on layout design

8.6.1 Corner Angles

If an outer contour is suitable for cut off according to the principle discussed in paragraph 8.4.1, it contains two parallel edges and two connecting edges. Preferably, the 'material containing' angle between a parallel and a connecting edge should not deviate too much from 90 degrees (figure 8.8). The corner where these edges meet should remain unrounded to avoid 'feather edges' /Bra-1, Bre-1/. A similar remark can be made for corner-angles between the parallel edges and notches in these edges. More in general the remark is valid for all corners in the outer contour for which both 'legs' will not be cut by the same punch. In contrast, corners for which both 'legs' will be cut by the same punch should always be rounded.
Figure 8.7: Practical example of a layout for a part that contains bends. Figure dimensions are 75% of original dimensions. Thickness for the part is 2 mm.
This design rule, commonly found in literature, has been derived considering tool life, ease of tool construction and risk of tool damage by 'feather edges'.

From the point of view of layout design the rule, and more in general corner-angles in the outer contour, pose problems. Accepting the supplied part-design as a given fact, as a normal process planning program would do, the rule has large consequences for major strip layout and tool layout choices. A rectangle with rounded corners would automatically lead to cut out for example, and rounded corners between notches and the edge in which they appear might lead to the rejection of normal cut off (figure 8.9).

In DIDACOE, where the part design is not assumed to be unchangeable, these and similar situations should at least lead to the generation of manufacturability information. Apart from this, the program can accept the consequences of corner-angles in generating a layout or it could simply ignore them. This will also depend on the question whether other influence factors permit cut off or not.

8.6.2. Tolerances and edge conditions

As shown in figure 8.10, blanking and piercing will produce burrs. Piercing (making holes) will produce burrs on the bottom side of the part while blanking produces burrs on the top side of the part (in the usual setup). The different types of final operations lead to different results here. It is one of the differences between One Part cut off and Bridge cut off for example. Two Part cut off will usually result in parts that are not equal to each other from the point of view of burrs. Thus, edge-conditions may influence strip layout design.

---

**Figure 8.8:** Corner-angles inside the cut-off line and between cut-off line and long strip edges. a = good, b = acceptable, c = not recommended because of 'feather edges' or unrounded corners in punch. (source: /Bre-1/, adapted)
Figure 8.9: Influence on layout design of corner-angles between notches and the edge in which they appear. Part (a) has 90 degree corners, which enables cut-off. Layouts (b), (c) and (d) are for the same part, but now corners have been rounded. Layout (b) is not acceptable due to the shape of the required piercing punch, (c) is good and (d) is not recommended (compare previous figure).

Using the cut off principle, parallel edges coincide with the strip edges and are not cut in the progressive die. One of the consequences is that a distance tolerance between the parallel edges can not be smaller than the strip width tolerance. If it is, the parallel edges have to be cut in the progressive die, for example by using cut out.

From the point of view of tolerances between features it makes a difference whether the features are produced in the same step of the die or not. Within the same step all tolerances are determined by the accuracy of the tool. Between two steps the accuracy of strip positioning (both in movement direction and perpendicular to it) plays an important role. From this point of view cut out and One Part cut off differ from the other final operation types. Consequently, the presence of small tolerances may influence strip layout choices.
Figure 8.10: Piercing or blanking will produce edges which are rounded on one side of the material and which have burrs on the other side. In the normal setup, piercing holes produces burrs on the underside of the part while blanking produces burrs on the topside. (Source: /Her-2/)

8.6.3 Other considerations

The strip layout influences efficient material use, the amount of scrap and consequently the material costs. From this point of view cut off solutions (especially normal cut off and two part cut off) are potentially better than cut out solutions. This is not always evident however. As noted in chapter 6 the cost estimation module assumes that the strip end is not used during cut off. In addition, notches in the parallel and connecting edges, or cut off bridges, may be so large that cut out provides a more efficient solution. Note that multirow solutions can be used to improve material utilization in cut out.

The strip layout also influences tool costs. In principle, cut off tools will be cheaper than cut out tools because total cut length is smaller in the first case. Note however that Two Part cut off requires most operations preceding the final step to be performed double. A choice for a multirow layout will also have a direct influence on tool costs, unless turn cutting is applied /Pic-2,Bla-1/.

Considering the calculation of operation times presented in chapter 6, the layout also influences operation costs. Pitch, strip dimensions, number of rows (tool multiplicity) and number of parts produced per strip all influence these costs.

Unfortunately, there will usually not be a solution for which all types of costs are minimal. Using a multirow layout for lower material costs will give higher tool costs for example. Probably operation costs will also go down, but theoretically this is not clear at first sight. Alternatively, optimum material utilization may require a large pitch, which is not optimum from an operation times point of view. This is the main reason that several promising layouts have to be generated, leaving it to a cost estimation module to choose the overall cheapest solution.
8.7 Edge Models and Advanced Features

8.7.1 Introduction

There are many factors that influence strip layout design. These factors interact and they often give contradictory indications. Therefore, the development of a complete layout design program from scratch is not an easy task. At this moment (January 1990) the prototype program implemented as part of DIDACOE does not consider all factors. The influence of bend lines and tolerances is completely neglected, and only part of the corner-angle problem has been handled. Most attention has been paid to the influence of the outer contour shape. The main 'reasoning mechanism' in the program is intended to recognize shapes that are suitable for cut off according to the principle described in paragraph 8.4.1. The same mechanism also allows the recognition of some specific shapes that will lead to specific layouts for cut out. The special cut off situations described in 8.4.2 have not been handled yet.

In the basic feature description the outer contour is described as a list of elementary basic features. From the point of view of 'reasoning' about the shape of the outer contour the latter features are not adequate. Of course they have to form the base for any reasoning (there is no other input), but higher level features that provide a more suitable description will be derived from them. These advanced features are all associated with Edge Models for the outer contour.

In general an Edge is defined as a part of the outer contour for which it is useful to distinguish it as a unit for reasons of strip layout design. It is described by a sequential list of outer contour elements (at least one) that together form an uninterrupted line from a start point to an end point. The direction of description of an edge is equal to that for the OC.
Strip layout generation consists of two steps. During the first step the program tries to define an edge model for the outer contour. In a second step layouts are generated considering the characteristics of the edge model.

8.7.2 Groups of LNL, Edge Holes and Straight Edges

The collection of all LIN-segments in the outer contour that lie parallel to each other is called an Angle Group of LIN (AGL). An OC will usually contain several AGL's. An AGL may contain several Line Groups of LIN (LGL's) which are collections of all LIN-segments in the AGL that lie on the same straight line.
Both AGL and LGL have a property called Suitability. A Line Group of LIN is defined to be suitable if all elements of the outer contour lie on or on the same side of the LGL-line. Suitability for an AGL is defined as the number of suitable LGL's that it contains. The value can be 0, 1 or 2 (figure 8.11a)

A suitable LGL is the base for a Suitable Straight Edge (SSE). This is an edge (see general definition above) that contains all LIN-segments of a suitable LGL, while the first and the last element in the edge element list (which may be the
Figure 8.11: Definitions of Angle Groups of LIN (AGL), Line Groups of LIN (LGL), Suitable Straight Edges (SSE) and Edge Holes (EHL). An LGL is suitable (suit=1) if all elements of the outer contour lie on or on the same side of the line defined by the LGL. Suitability for an AGL is the number of suitable LGL’s it contains. A suitable LGL is the base for definition of a SSE (b). Nett length for a SSE is \(a+c+d+e+f\). Total length is \(a+b1+c+d+e1+f\). Combined length is \(a+b2+c+d+e2+f\). EHL’s have opening lengths (b1 and e1).

same element) are LIN-segments from this LGL. The line defined by the LGL is the 'baseline' for the edge.

If a SSE contains several LIN-segments, they can be separated by Edge Holes (EHL’s). An EHL is an area outside the outer contour that is bounded by a part of a suitable straight edge (the EHL-contour) and an imaginary LIN-segment between the start and end points of the EHL-contour. The following conditions have to be met (figure 8.11b):

- the EHL-contour consists of one element of the SSE or of a list of SSE-elements that together form an uninterrupted line,
- the start point of the first element in the EHL-contour coincides with the end point of a LIN-element on the SSE-baseline,
- the end point of the last element in the EHL-contour coincides with the start point of a LIN-element on the SSE-baseline,
- the EHL-contour does not contain any LIN-segment that belongs to the base-LGL for the SSE.
An opening length is defined for an EHL. It is the distance between start and end point of the EHL-contour: the length of the imaginary LIN-segment. Four lengths are defined for an SSE:
- nett length: the sum of the lengths of LIN’s in the base-LGL,
- total length: the sum of nett length and all EHL-opening lengths,
- combined length: the sum of nett length and all EHL-contour lengths,
- intersection length (to be defined later).

8.7.3 Groups of RAD, Circle Interruptions and Circle Edges

A collection of all RAD’s in the outer contour that lie on the same circle is called a Circle Group of RAD (CGR). The property suitability is also defined for CGR’s. A CGR is considered to be suitable if all elements of the outer contour lie on or inside the circle defined by the CGR. Although not very likely, an OC may contain more than one suitable CGR.

A suitable CGR is used as the base for a Circle Edge if the RAD-segments in the CGR cover more than half of the circle. More formal: the sum of the lengths of the RAD-segments in the CGR has to be larger than half of the length of the circumference of the circle defined by the CGR (figure 8.12). In any OC, only one CGR can meet this condition. In contrast with a SSE, a Circle Edge (CE) describes the complete outer contour. It consists of all RAD-segments in its base-CGR and of Circle Interruptions (CINT’s) that ‘fill the gaps’ between these RAD-segments.

A CINT is an area outside the outer contour that is bounded by a part of this contour (the CINT-contour) and by an imaginary RAD-segment that lies on the base-CGR circle between the start and end points of the CINT-contour. The following conditions have to be met:
- the CINT-contour consists of one element of the Circle Edge or of a list of CE-elements that together form an uninterrupted line,
- the start point of the first element in the CINT-contour coincides with the end point of a RAD-element that belongs to the base-CGR for the CE,
- the end point of the last element in the CINT-contour coincides with the start point of a RAD-element that belongs to the base-CGR for the CE,
- the CINT-contour does not contain any RAD-segment that belongs to the base-CGR for the Circle Edge.

An opening angle is defined for a CINT. This is the angle between start point of the CINT-contour, centre point of the base-CGR-circle, and end point of the CINT-contour. It follows from the definition of a circle edge that this angle is always smaller than 180 degrees. A nett length, total length and combined length are defined for a CE similar to those for a SSE.

8.7.4 Straight Edge Model creation without suitable AGL

Whether or not the program creates straight edge models depends on the suitability of AGL’s:
1) if there is at least one AGL with suitability 2 (implying that it contains two suitable LGL’s), the strategy described in the next paragraph is applied,
Figure 8.12: Definitions of a Circle Group of RAD (CGR), Circle Edge (CE) and Circle Interruptions (CINT). A CE is created only if the RAD-segments cover more than half of the circle circumference: \(a+c+e\) larger than \((a+b+c+d+e+f)/2\).

2 if there is no AGL with suitability 2, but there are at least two AGL’s with suitability 1, the strategy described in this paragraph is applied.

3 otherwise no straight edge model is generated.

The second case above implies that the outer contour contains two or more suitable LGL’s that all belong to different AGL’s. That is: no two lines defined by the LGL’s are parallel.

Following the outer contour in its direction of description, these LGL’s will be encountered in some sequence. The program will start its analysis by defining GAP’s between any two consecutive LGL’s.

A GAP is a temporary abstract feature defined by two suitable LGL’s, the one that comes first in the sequence (FLGL) and the one that immediately follows it (SLGL). As the lines defined by these LGL’s are not parallel, they intersect (figure 8.13). The intersection point can be ‘near’ the end point of the FLGL and the start point of the SLGL (the normal case), or ‘near’ the start point of the FLGL and the end point of the SLGL (the abnormal case). In the normal case a Gap Angle is defined as the smallest angle between the two LGL-lines. A first Gap Length and a second Gap length are defined as shown in figure 8.13. In the abnormal case the Gap Angle is defined as \(2\pi - \Pi\) minus the smallest angle between the two LGL’s, leading to a value larger than \(\Pi\). In this case Gap Lengths are not defined.
Figure 8.13: Definitions of GAP's between two consecutive suitable LGL's. GAP-angles (GA's) smaller than 180 degrees are obtained when the intersection point of the LGL's is near the endpoint of the first LGL (FLGL). Such angles will lead to creation of Corner Holes, unless both Gap-Lengths (GL1,GL2) are zero. For GAP-angles larger than 180 degrees Other Edges will be created. Note the directions of the LGL's!

Figure 8.14: Definition of a Corner Hole (CHL). A CHL is defined near the intersection of two SSE's when the associated GAP-angle is smaller than 180 degrees unless both GAP-lengths are zero (see previous figure).
After completion of the GAP-creation the program builds the edge model. Every suitable LGL leads to the definition of a suitable straight edge, which may include edge holes. GAP's with an angle smaller than $\pi$ lead to definition of Corner Holes unless both Gap Lengths are zero, while GAP's with an angle larger than $\pi$ lead to the definition of Other Edges.

An Other Edge (OE) is a part of the outer contour that is not a suitable straight edge and that provides a connection between two SSE's that meet one of the following conditions:
- they are parallel,
- the Gap Angle defined by their baselines is larger than $\pi$.

A Corner Hole (CHL) has the same definition as an Other Edge, except that the Gap Angle defined by the baselines for the two connected SSE's is smaller than $\pi$. Similar to an Edge Hole, it can be described as an area outside the outer contour that is bounded by a part of the outer contour (the CHL-contour) and two imaginary LIN-segments as explained below. The following conditions have to be met (figure 8.14):
- the CHL-contour consists of one outer contour element or of a list of outer contour elements that together form an uninterrupted line,
- the start point of the first element in the CHL-contour coincides with the end point of a LIN-segment that is the last element in a SSE,
- the end point of the last element in the CHL-contour coincides with the start point of a LIN-segment which is the first element in another SSE,
- the imaginary LIN-segments referred above lie between the start point of the CHL-contour and the intersection point of the two SSE-baselines, and between this point and the end point of the CHL-contour. The segment lengths equal the Gap Lengths defined above.

A typical example of an edge model created in this way is presented in figure 8.15a. The model will contain three edges or more.

8.7.5 Straight Edge Model creation with suitable AGL

If the outer contour contains at least one AGL with suitability 2, the approach to edge model creation is slightly different. The program will now first select the AGL from which edge model creation starts. If there is only one AGL with suitability 2, this is no problem. If there are more, the program will select the AGL with the largest total length of suitable LGL's (LGL total length is defined similar to SSE total length).

The selected AGL contains two suitable LGL's, that have a program-internal administration number. The LGL with the lowest number leads to the creation of Edge 1, the other to the creation of Edge 3. Both are suitable straight edges that may contain edge holes. Edges 1 and 3 potentially are the parallel edges required in the cut off principle.
Next the program tries to complete the edge model by defining the connecting edges, numbered 2 and 4 (in OC direction of description). The same approach is followed for each of these edges separately:

- if the connecting part of the OC does not contain a suitable LGL, an Other Edge is created,
- if the connecting part contains one suitable LGL, a SSE is created from it. This SSE may include edge holes. Corner holes are defined near the intersections of this edge with the edges 1 and 3, if necessary.
- if the connecting part contains several suitable LGL's, the one with the largest total length is used to define SSE, EHL's and CHL's.

This approach always provides a four-edge model. A typical example is presented in figure 15b.
8.8 Edge Relations

8.8.1 Uniformity

As indicated in paragraph 8.4.1, connecting edges have to 'match' in some way to enable Normal or Two Part cut off. Matching of edges is dealt with by introducing the concept 'edge uniformity'. For the time being uniformity is only defined for opposite edges in a four-edge model if the two other opposite edges are parallel.

Three main types of uniformity are distinguished: normal, mirrored and rotated uniformity (figure 8.16).

Two edges are normal uniform if they coincide completely after a translation of one of the two over a suitable distance parallel to the parallel edges. The suitable distance is the distance between the two edges measured along any line parallel to the parallel edges.

They are mirrored uniform if they coincide completely after mirroring one of the two edges in the line middle between the two parallel edges first, and then performing the suitable translation.

Rotated uniformity is a property of an edge itself and not a relation between edges. An edge is assumed to be rotated uniform if it coincides completely with itself after a 180-degree rotation around the intersection point of the edge and the line middle between the two parallel edges. The associated relation between two opposite edges is 'both rotated uniform'.

![Figure 8.16: Imaginary operations to check (a) normal-, (b) mirrored- and (c) rotated-uniformity of connecting edges in a four-edge model with two parallel edges. The line middle between the parallel edges is the line for mirroring. The intersection point of this line with an edge is the centrepoint for 180 degree rotation.](image-url)
Unfortunately it is not sufficient to distinguish only between completely uniform edges and not uniform edges. As shown in figures 8.3 c,d and e cut off can also be advantageous if the match between the edges is not complete. The concept ‘interrupted uniformity’ is introduced to reflect these cases. It is defined in combination with normal, mirrored as well as rotated uniformity.

In general, two opposite edges are conceived to be interrupted uniform if they coincide partially after the translate, mirror and/or rotate operations indicated above, while the non-coinciding parts define potential areas in the strip that can be punched out prior to the final operation. A more formal definition for interrupted normal uniformity is given below.

Assume that a blank has been placed in the strip coordinate system and that it has been rotated in such a way that the parallel edges coincide with the long strip edges. Any line in the strip that runs parallel to the parallel edges (YS is constant) will intersect both connecting edges. The two intersection points define the distance between the two edges for this YS. It is possible that one or more ranges of YS-values exist for which this distance is constant and maximal. These ranges represent parts of the edges that will coincide after a translation of one of the edges towards the other parallel to the parallel edges over a distance equal to this maximum distance. Potentially, these edge parts form the future cut off line. YS-ranges in which the distance is less than maximal represent non-coinciding parts of the edges that define future areas in the strip that can be punched out in steps prior to the final step.

The edges are defined to be interrupted normal uniform if they are not completely normal uniform, while there is at least one YS-range (with a predefined minimal length) for which the distance between the edges is constant and maximal.

The definition can be adapted for cases where a line YS-constant has more than one intersection with an edge. Similar definitions are possible for interrupted mirrored uniformity and, with some changes, for interrupted rotated uniformity.

The examples of interrupted uniformity presented in figures 8.3c,d,e show that the concept can be useful. Figure 8.17 shows some situations in which the connecting edges are interrupted uniform according to the above definitions, while cut off is doubtful at least. Problems arise near the points where ‘coinciding’ and ‘non-coinciding’ parts of the edges meet. These problems are directly associated with the corner-angle problem discussed in paragraph 8.6.1. In this case there are two angles of interest:

- in each of the edges separately: the ‘material containing’ angle between tangents to the edge at the end of a coinciding part and at the start of the immediately following non-coinciding part (A1, A2 in figure 8.17). This angle should not be too small to avoid ‘feather edges’.

- the angle between the tangents to both edges at the start or end of a non-coinciding part of the edge (B in figure 8.17). This angle is important because it describes a ‘corner’ in the punch that will be used to pierce the future strip area described by the non-coinciding parts. Preferably the two tangents should coincide, which implies that there is no corner in the punch, or that the corner is rounded.

132
Figure 8.17: Interrupted uniform edges are not always suitable for cut-off with preceding piercing of holes or notches. Angles A1 and A2 should not be too small to avoid 'feather edges'. Angle B should preferably be 180 degrees. Arrows point to corners that will give problems.

These ideas have only been detailed for suitable straight edges for which the baselines coincide after the imaginary transformation(s) for uniformity check described above. As explained before, imaginary LIN-segments associated with EHL's and CHL's can lie on the baselines of such edges. The above problems arise if the start and/or end points of imaginary LIN-segments of one edge lie somewhere in the middle of a 'real' LIN-segment of the other edge, when baselines are imagined to coincide. Consequently it is favourable if CHL's and EHL's of both edges match in some way (coinciding start and end points).

In addition to the earlier conditions, two SSE's are considered interrupted uniform only, if their CHL's and EHL's match according to one of the examples presented in figure 8.18. If they would be interrupted uniform according to the original definition while CHL's and EHL's do not match, the SSE's are considered doubtful uniform. The program manager can influence program behaviour on this point by predefining whether doubtful uniformity is 'enabled' or 'disabled'. In the first case it is treated as interrupted uniform, in the latter case it is treated as not uniform.

Even with this extended definition, interrupted uniformity for SSE's does not guarantee suitability for Normal or Two Part cut off: corner-angles should be checked even if CHL's and EHL's match!
Figure 8.18: Examples of CHL/EHL-match that allow interrupted uniformity of suitable straight edges. (a) CHL matches other CHL, (b) CHL matches CHL and one or more EHL's, (c) CHL matches EHL, (d) EHL matches EHL, (e) EHL matches several EHL's in opposite edge. There should not be any start- or endpoint of an 'imaginary' LIN-segment that lies on a 'real' LIN-segment in the opposite edge when SSE-baselines coincide.

8.8.2 Angles and intersection lengths

As explained above, an outer contour can be conceived as a sequence of edges. Two consecutive straight edges may be separated by a corner hole or by an Other Edge. A number of edge properties have been defined to describe relations between consecutive edges.

The intersection length for a suitable straight edge is defined as the distance between the points where the preceding and the succeeding edge 'intersect' its baseline. These intersection points coincide with the start or end points of the edge, or they are the intersection points of the imaginary LIN-segments of the CHL's. Consequently, the intersection length for a SSE can also be defined as the sum of its total length and the lengths of the imaginary LIN-segments of CHL's that lie on its baseline.

Related to the corner-angle problem, a Start Difference Angle (SDANG) and an End Difference Angle (EDANG) are defined for every SSE and OE. The SDANG for an edge is defined as the angle inside the outer contour (the 'material containing' angle) between the tangent to the edge in its start point and the tangent to the preceding CHL or edge in its end point. The EDANG is defined in a similar way. Similar angles are defined at the start and end of EHL's.
8.9 Layout creation

8.9.1 Layouts for cut off

For the time being only a four-edge model can lead to the generation of a strip layout for cut off. Two main types of layout are distinguished: PARAL13 and PARAL24.

A PARAL13-layout is generated if edges 1 and 3 are parallel, while bend lines, tolerances and corner-angles do not prevent a coincidence of these edges with the long strip edges. (For the time being, bend lines and tolerances never prevent layouts of any kind.) A PARAL24-layout is generated under similar conditions, when edges 2 and 4 are parallel and can coincide with the long strip edges. Depending on the situation neither, one or both of the layouts can be generated.

If the PARAL13 and -24 layouts are possible, different subtypes of layouts can be produced, associated with the different types of cut off. This will only be explained for PARAL13. For PARAL24 it is completely similar. PARAL13 layout creation is governed by the following rules:

1. if edges 2 and 4 are assumed normal uniform while their distance tolerance does not require production in the same step of the die, a one-row layout is generated for normal cut off,

2. if edges 2 and 4 are assumed mirrored uniform but not normal uniform, while distance tolerance for edges 2 and 4 does not pose problems, a two-row layout for Two Part cut off is generated with blanks in the second row mirrored in the centre line of the strip,

3. if edges 2 and 4 are assumed both rotated uniform but not normal or mirrored uniform, while tolerance does not pose problems, a two-row layout for Two Part cut off is generated with blanks in the second row rotated 180 degrees,

4. if edges 2 and 4 are not assumed uniform in any way, while distance tolerance does not pose problems, a one-row layout for Bridge cut off is generated.

5. if a tolerance on the distance between edges 2 and 4 requires production of both edges in the same step, a one-row layout for One Part cut off is generated.

Note that edges are assumed uniform if they are completely uniform or if they are interrupted uniform, and, depending on settings, if they are doubtful uniform.

8.9.2 Layouts for cut out

A layout for cut out is generated only if the above layouts for cut off are not possible. The type of layout that will be generated depends on the perimeter type (PMTYP) of the outer contour. Some special perimeter types with associated layouts have been defined. If the outer contour is classified as belonging to one of these types, the special predefined layout solutions are used, otherwise the default strip generation procedure is applied (next paragraph). The special perimeter types will now be discussed first. Their currently associated layouts are shown in figure 8.19. It is easy to change these layouts or to add others.
Figure 8.19: Current standard layouts for cut-out for outer contours that are classified as belonging to one of the defined perimeter types. Example only: it is easy to change the layouts or to add alternative layouts.
The outer contour is considered to be a 'rectangle' (PMTYP=1) if:
- a four-edge model exists in which both edges 1 and 3 and edges 2 and 4 are parallel. This implies that all edges are suitable straight edges,
- edges 1 and 3 are perpendicular to edges 2 and 4,
- for all edges, the total length is longer than half of the intersection length.
This restricts the size of corner holes.

Two layouts are created for a rectangle: one with edges 1 and 3 parallel to the long strip edges and one with edges 2 and 4 parallel to the long strip edges.

If the same conditions are met, but edges 1 and 3 are not perpendicular to edges 2 and 4, the outer contour is classified as a 'parallelogram' (PMTYP=3). This leads to the same layout modes.

The QC is treated as a 'rectangle with one slant edge' (PMTYP=2) under the following conditions:
- a four-edge model exists in which edges 1 and 3 are parallel, while both edge 2 and edge 4 are suitable straight edges,
- either edge 2 or edge 4 is perpendicular to edges 1 and 3, but not both,
- for all edges, the total length is longer than half of the intersection length.

This situation leads to two layouts. The first layout has two rows where blanks in the second row have been rotated over 180 degrees. The other layout has one row with the slant edge parallel to the long strip edges.

If the same conditions are met, but neither edge 2 nor edge 4 is perpendicular to edges 1 and 3, while edges 2 and 4 are not parallel, the outer contour is classified as a 'trapezoid' (PMTYP=4). This leads to three layouts. The first has two rows with edges 1 and 3 parallel to the long strip edges and blanks in the second row rotated over 180 degrees. The other layouts have one row with respectively edge 2 and edge 4 parallel to the long strip edges.

An outer contour is treated as a 'triangle' (PMTYP=5) if:
- there are exactly three non-parallel Suitable Straight Edges,
- there is no Other Edge,
- for all edges the total length is longer than half of the intersection length.

Three layouts will be generated for a triangle. Every layout puts a different edge parallel to the long strip edges. All layouts have two rows where blanks in the second row have been rotated over 180 degrees.

An outer contour is treated as a 'circle' (PMTYP=6) if it consists of one element of type CIR, or if a Circle Edge has been created while one of the following conditions is met:
- there are no Circle Interruptions in the edge, or
- for every CINT in the edge the opening angle is smaller than a pre-defined value.

A one-row layout and a three-row layout are generated for circles. In the latter, blanks are placed in such a way that material utilization is optimal.

If the outer contour has two parallel edges 1 and 3 while it can not be classified as one of the preceding perimeter types, it is assigned PMTYPR-9 for 'other parallel'. Two one-row layouts are created, one with the parallel edges parallel
to the long strip edges and one with the former edges perpendicular to the latter.
In all other cases the OC is of type zero: 'default'.

89.3 Pitch determination for non–trivial cases

In the special cases discussed above the orientation angle is derived directly from the outer contour shape. Usually, values for the pitch and for other required output variables are easily calculated from edge lengths and corner hole dimensions, taking into account bridge widths, orientation angles and mirroring. These details will not be explained here.
There are three cases in which determination of the pitch poses problems:
- for Bridge cut off and One Part cut off when the connecting edges are not both suitable straight edges,
- for PMTYP-9 ('other parallel') when the parallel edges are parallel to the long strip edges,
- for PMTYP-0 ('default'). In this case the orientation angle is also unknown.
For any orientation angle the pitch is determined by generating an offset line on the outside of the outer contour at a distance of half the bridge width. For cut out and for One Part cut off this bridge width is determined from BWOOF * T, where BWOOF is a material dependent bridge width factor (typically 1.5) and T the strip thickness. For Bridge cut off twice this value is used. For the time being bridge length and material properties do not influence bridge width determination.
The pitch is now determined in such a way that offset lines from two adjacent blanks in the strip touch each other in at least one point without intersecting each other anywhere. The implemented procedure can handle one-row layouts and two-row layouts where blanks in the second row have been mirrored or rotated as described before.
The offset line is first rotated to the desired orientation angle in the strip. For any rotation it is possible to define two sides of the offset line: one clockwise, the other anti-clockwise, from the point with the maximum YS-value going down along the offset contour to the point with the minimum YS-value (figure 8.20). Both sides contain a point with maximum YS-value, a point with minimum YS-value, and potentially start/end points of offset contour segments and points where the tangent to the contour is described by a line YS-constant. All these points have YS-coordinates that can be sequentially ordered in a list. This permits the definition of YS-intervals between any YS in the list and the next lower YS.
Every interval is examined separately. The objective is to find the largest distance in every interval between two points from different sides that have the same YS-coordinate. This distance represents the smallest possible pitch from the point of view of the interval. It will be found either for the highest or lowest YS-value in the interval, or for a YS for which the tangents to both sides are parallel /Bla-1/. In some difficult situations the distance between both sides is determined for a large number of YS-values in the interval. In this way the program can also handle situations where both sides contain three points with the same YS-value, while the sides 'fit into each other'. The largest distance found after examination of all intervals is the desired pitch.
Figure 8.20: Definition of Sides and YS-intervals for use in pitch-determination for default cut-out cases. The point with maximum YS, the point with minimum YS, start/endpoints of segments and points where the tangent to the outer contour is described by a line YS-constant, define the YS-values that bound the intervals. The maximum distance between the two sides in an interval represents the minimal possible pitch from the point of view of the interval.

For the 'default' cut out case this procedure is applied for a predefined large number of orientation angles. Pitch, minimal strip width and number of blanks per strip are determined for each angle. The number of blanks is determined from SL/PITCH where a predefined value for the strip length SL will be used. A temporary material use percentage can now be derived from:

\[
MU\% = \frac{(\text{Outer Contour Area} \times \text{Number of blanks per strip}) \times 100}{(\text{Minimal Strip Width} \times \text{Strip Length})}
\]

For the time being, only the orientation angle which leads to the highest material use percentage will be selected for output.

8.10 Program evaluation

Although the strip layout generation program is a prototype which neglects a lot of influence factors, it is already surprising what it can do: the edge model concept works conveniently. On the other hand it is sometimes even more surprising what the program can not do yet. The general impression is however that most of the basic work has been done and that a lot of the current problems can easily be solved. Among these are:
- complete solution of the corner-angle problem,
- improvement and extension of the uniformity checks,
- increasing the number of special cases to be recognized,
- improvement of the layouts to be generated in each specific case.
Some other problems will require more work. The most important ones are:

- the current definition of suitable straight edges, although useful, poses a problem. As shown in figure 8.21 identical connecting edges can lead to different edge models, depending on the side of the edge where the material is. This makes the recognition of uniformity more difficult. It may be advantageous to define other types of straight edges that have 'protrusions' instead of edge holes. This may lead to a situation where alternative edge models exist for the same part of the outer contour.

- in the special cases, layouts are generated based on the recognition of some basic outer contour shape. Even if this basic shape is distorted by corner holes and edge holes, layout generation strategy often remains the same. For large CHL's and EHL's this will sometimes lead to bad or even ridiculous results. All information to restrict sizes of CHL's and EHL's is present, but the number of possible shapes for CHL's and EHL's and the number of combinations of these distortions is so large, that it is difficult to find useful general rules. Too stringent conditions will prevent the recognition of possibilities for advantageous layouts, but on the other hand the need for restrictions is clear.

- the influence of bend lines on layout generation has two aspects. It will probably not be difficult to take required angles between bend lines and grain direction into account. More difficulties will be associated with the definition of the typical layouts for parts containing bends (figures 8.6b, 8.7). A first problem is to make the program decide which face will remain in the plane of the strip (the reference face). The next step is to determine bridge widths between blanks and between blanks and strip edges in such a way that sufficient width remains after piercing of the holes required to free the faces.

- at the moment the program can only handle very special multirow layouts. Especially for the default cut out case it would be desirable to have more freedom in generating multirow layouts. Rules have to be found that indicate when such layouts can be interesting. In addition the brute force angle variation procedure should be extended to handle general multirow layouts. This will not be easy.

![Diagram](image)

**Figure 8.21:** Uniform connecting edges may have different edge-model representations depending on the side of the edge where the material is.
The prototype program does not supply manufacturability information yet. Even with the current limited implementation it should not be too difficult however to generate such information. Possible topics that could already be handled now are:
- corner-angles in the outer contour in general and more in particular between edges and between notches and the edge in which they appear,
- size and position of corner holes and edge holes. If suitable straight edges are doubtful uniform the program should be able to indicate how they can be made to be interrupted identical (matching CHL’s and EHL’s).
- suggest changes in the design in XS- or YS-intervals that require a relatively large strip width or pitch as compared to other intervals.

8.11 Notes on the development of a tool layout design program

Tool layout design has not been studied in detail yet. This is mainly due to lack of time and to lack of a method for tool cost estimation. The input required by such a method determines to a considerable extent the information that a tool layout design program should supply.
For the time being it is imagined that the output of a tool layout design program should include:
- type of combination of strip positioning aids,
- a description of the contours of all working openings in the cutting plate and their positions and orientations in the strip coordinate system, assuming that the first blank in the strip is positioned under the last step in the die,
- actual strip width,
- dependent variables like total cut length, total bent length and number of steps in the die.

A tool layout program will first have to decide which type of combination of positioning aids is to be used ('french' side trimming stop, fixed stop, side stops, and the like). Related to this are the decisions whether pilots should be used or not, whether existing holes can be used as pilot holes, and if not, where new pilot holes (outside the blanks) can be created.
In a next step the program should make a list of all potential partial operations that are to be performed in the die. This is a list of all holes and bends to make. For this purpose, the program should define which faces constitute one bend. The list of holes will contain all holes present in the blank description. In addition the program will have to define new holes that are not defined in the blank description yet because they describe areas outside the outer contour. The latter holes include:
- holes required to cut free faces that have to be bent,
- edge holes between the outer contour and the long strip edges,
- (combined) edge holes between two blanks in the strip,
- (combined) corner holes between one or two blanks and a strip edge,
- pilot holes and 'holes' associated with side trimming stops,
- other similar holes.
Every hole and every bend represents a potential operation. The program will have to decide for every potential operation separately if it will be performed in one step or in several steps. It is quite possible that holes with complex contours will be produced in several steps using two or more punches. If hole dimension tolerances require it, piercing in one step may be followed by one or more shaving operations in the next steps. Similar decisions have to be made for bends.

This phase of the program will generate a list of real partial operations from the list of potential ones.

In a third step the program will have to decide which partial operations are to be performed in which step in the die. The main guideline will be to punch holes first, to form bends next and then to apply the final cut out or cut off operation. In addition the program will consider distances between holes. Two holes that are too close together can not be pierced in the same step. In contrast, small distance tolerances may require the simultaneous production of two features in the same step. Considering tool construction demands the program will also have to decide whether idle steps are required or not.

It should not be too difficult to generate the list of potential operations. Converting this list into a list of real partial operations requires a considerable amount of knowledge and experience. Probably, this is the most difficult phase. The third part of the program, planning partial operations per step, may be slightly easier again, although knowledge of tool construction demands is required.
9 The Knowledge-Based System experience

The strip layout design program has been implemented as a Knowledge-Based System (KBS), using Artificial Intelligence (AI) techniques. This chapter introduces the concepts AI and KBS and evaluates the first experiences in working with AI-techniques.

9.1 A short introduction to Artificial Intelligence

9.1.1 What is AI?

Many different definitions of the concept Artificial Intelligence (AI) can be found in literature /Wie-8/ but none of them seems to be generally accepted. The description of AI that follows has been inspired by the definition supplied in /Ame-2/.

Artificial Intelligence can be conceived as a field of science that studies the behavioural aspects of human thinking, learning and problem solving, with the objective to simulate this behaviour in computers. Unfortunately, AI is not always used in this sense. Very often the term AI simply refers to a set of techniques that are used in computer programs to solve problems in another way than in conventional programs. These AI-techniques do not replace algorithmic programming techniques, but are an addition to them. AI-techniques are better (more convenient) than conventional techniques for some types of problems only. For other types the algorithmic approach is still the best solution.

Although AI-techniques are a result of AI as a science, it is not necessary to associate them with anything human or intelligent. From an engineering point of view it is only confusing to do so. It creates an atmosphere of mysticism, leads to the difficult question 'what is intelligence?' and starts the philosophical debate on whether machines will ever behave like human beings. It is much more practical just to conceive them as a useful set of techniques that can be used in computer programs to solve problems, and to explore their possibilities.

9.1.2 The main AI-techniques

It is impossible here to explain all AI-techniques in detail and to discuss their specific usefulness. Only a very brief introduction to the main principles is given here. For more extensive information the reader can refer to /Ric-2,Win-1,Her-4/ or to the summary in /Wie-6,10/.

In general a problem can be conceived as the task to find a path from some initial state to some desired final state. This path will usually contain several intermediate states. The possible states, their sequences in a path and all possible paths are defined by a state space (figure 9.1). It is useful here to think of chess, where the start position of the pieces on the board is changed by moves until a checkmate situation is obtained.
Three things are needed to describe and solve a problem. First it should be possible to describe the states: the collection of known data at any moment. Second the valid operators that can be applied to change a state into another state have to be defined (the valid moves in chess). In any state it is usually possible to apply different operators. Not all operators will lead to a state that is closer to the final desired one. In addition some operators will be more difficult to apply, or have more negative side-effects (higher 'costs') than others. This implies that strategic knowledge is required: it is necessary to know which operator is probably the best in a given state.

![State-Space Diagram]

**Figure 9.1:** A problem can be conceived as the task to find one path, all paths or the best path from some start state (S) to some goal state (G). Such paths will contain several intermediate states (A–F). The State-Space defines the possible states and the sequences in which they can appear in a path. The lines that connect the states represent operators that can change a state. The numbers next to the line are weights or 'costs' of the operators. The second part of the figure shows all possible paths from S to G in forward chaining (reasoning from start position to goal position). Several search-strategies have been developed in the AI to find desired paths in such search-trees as fast as possible (Source: /Her-4/).
A state could be described by an unordered collection of facts (bishop1 is black, bishop1 is on field C3, bishop1 can not move). Often it is beneficial to consider related facts together as one unit. This unit is typically called an Object, which has a name and attributes that describe its relevant characteristics. It would be possible for example to create an object with the name 'Bishop1' and attributes 'PieceColour', 'FieldColour', 'Position' and 'Deployment'.

The state description can be enhanced by defining relations between objects. Different types of relations can be imagined: 'is part of', 'is made by', 'is used for' and the like. In the chess example it might be useful to store that 'Bishop1 attacks Bishop3', or that 'Bishop1 protects Horse2'.

A state description will often contain several objects of the same type. To avoid that the same attributes have to be defined for every object separately it is advantageous to define Classes. A Class is a template for a type of objects. Real objects are instanciaed from their class, thus inheriting all attributes and other characteristics that have been defined for that class. Objects have an 'is a'-relation to their class.

Classes can be ordered in a hierarchy (figure 9.2). In such a structure, any class can have parent classes that represent generalizations and child classes that represent specializations. The Class 'Bishops' could have a parent class 'FastMovingPieces', which in its turn would belong to the class of 'Pieces'. Classes that are a specialization of another class will inherit all characteristics from that parent class. In addition they may have some other properties. Class relations are of the type 'is a kind of'.

Classes can be used to define more than just attributes. Information can be attached to each attribute to specify how its value can be determined in a specific state. Similar information could be used to specify how the value of the attribute is allowed to change, or when it changes. It would be possible for instance to specify in the class 'Bishops' how the value of the attribute Position can be altered (how the pieces of this type are allowed to move). This type of information would also be inherited by the instances of the class. (See /Mcc-1/ for the related topic of object oriented programming).

In addition the existence of classes allows the definition of general rules. Thanks to a class it is possible to write 'Bishops always stay on fields of the same colour' instead of having four such rules for every object Bishop separately.

Thus, objects and classes are used to describe the states, and they can be used to specify allowable operators or promising strategies. Another way to specify methods for attribute value determination, allowable operators and best strategies is offered by Production Rules. Such rules are of the type: 'If (conditions) Then (actions)'. Typically the actions will be conclusions, in the sense that the value of some boolean variable is proven to be 'true' or 'false'. This conclusion may lead to some associated actions that change the state or enhance the state description. A program that uses AI-techniques usually contains several hundreds of rules. Rules are separated units of knowledge that have no direct relation to each other, although it is possible that rules associated with the same topic are grouped together in a context, to structure the knowledge base in which they are stored.
Rules and Objects are knowledge representation facilities. A program needs an additional facility that knows how to use these units of knowledge. This facility is usually called the inference engine. It uses some search strategy that is suitable for the type of problem to be solved. Different strategies have been developed.

Starting from some known state the program can search for rules whose conditions are met. If such a rule is found, its conclusion is confirmed and associated actions can be performed. This leads to new knowledge that may enable the 'firing' of other rules, until the desired state is reached. This mechanism is called forward chaining (figure 9.1).

Alternatively the program may start from the desired state and search for rules whose conclusions lead to this state. If such a rule is found, the program will check if its conditions are met. In doing so it will need to evaluate other rules, that have conclusions that are relevant to prove the conditions of the first rule. This backward chaining mechanism will continue until all conditions of a rule can be proved based on the description of the initial state.

At any moment in both mechanisms there will probably be several rules that are interesting to explore next. Considering this, two different strategies can be distinguished. In breadth first search the program will first try all possibilities that offer itself at a certain moment (A and D in figure 9.1). In depth first search it will select one of these possibilities, execute it and then select one possibility in the new situation (A, B, C in figure 9.1). Different theories have been developed to limit these search processes or to make them more efficient. It is important to note that the best search strategy depends on the type of problem. It makes a lot of difference for instance if one path, the best path or all paths from initial to desired state have to be found.
Other AI-techniques are associated with handling vague, uncertain or contradictory data. Heuristic data (experience knowledge) are often of this type. These techniques will not be explained here. Note however that the use of the probability mechanisms is not undisputed. Pamas /Par-1/ points out for example that these mechanisms do not lead to the trustworthy programs needed in engineering.

9.1.3 What is a KBS

A Knowledge-Based System (KBS) can be conceived as a computer program that uses AI-techniques to perform its tasks. In addition most definitions demand that a KBS consists of at least four components:

- a knowledge base containing the units of knowledge (objects and rules),
- an inference mechanism that incorporates the problem solving strategy (the search strategy),
- an explanation facility that can inform the user of the program on why a question is being asked and on how a conclusion was reached,
- a knowledge acquisition facility that allows the programmer to add knowledge to the system without reprogramming or recompiling it.

The number of books on Knowledge-Based Systems or Expert Systems (there hardly seems to be a difference between the two) is overwhelming. General information on the structure of such systems and on their development can be found for example in /Dym-1,Nijh-1,Hay-1,Swa-1,Wie-6/.

9.1.4 When use AI-techniques?

There are no clear rules that explain when conventional programming should be used and when AI-techniques offer advantages. Only a number of often vague indications can be given. The main reason for this is that the question to ask is not whether the problem can or cannot be solved using AI-techniques or conventional algorithms. The question is which of the two will be most suitable. Often both solutions will be possible, and the programmer has to compare them to judge which one will work most conveniently. This, of course, requires experience in both.

Some indications can be obtained by looking at the types of problems for which knowledge-based systems have been developed. The standard list encountered in literature usually contains: Analysis Assistance, Interpretation, Diagnosis, Planning, Design, Repair and Maintenance, Monitoring and Control and Instruction /Dym-1,All-1,Wie-6,10/. Especially diagnosis and interpretation problems are said to be suitable for implementation in a KBS. Literature also supplies some checklists for the use of AI-techniques (see /Mer-1/ for example). The best help however is to understand the fundamental differences between a conventional program and a KBS.

The main difference between a KBS and a conventional program is probably that the latter uses algorithms that completely fix the program actions and their sequence of execution. These algorithms contain both the units of knowledge
and the problem solving strategy as an integrated whole. A KBS clearly separates these two elements in the knowledge base and the inference engine. The sequence of actions is determined at runtime. This difference has a number of consequences:

- the conventional programmer has to understand the problem and its solution in all details. This limits the complexity of the problems that can be handled in a reasonable time. To some extent, the development of a KBS requires less insight in the overall problem. The programmer can concentrate on the definition of separate pieces of knowledge and on the definition of the general solution strategy. This potentially enables the programmer to handle more complex problems and problems that are not completely understood or not well structured yet /Esm-1, Wit-1/;

- due to the mixture of knowledge and problem solving strategy, it is difficult to change or extend conventional programs. It is comparatively easy to change or add units of knowledge in a KBS without reprogramming or recompiling and without adapting the solution strategy. This implies that AI-techniques are potentially useful in fast prototyping,

- a conventional program typically has a fixed type of input and a fixed type of output, both predefined by the programmer. Potentially a KBS can be used in a more flexible way. The user can volunteer different types of input and ask the program to answer different types of questions. The KBS can be said to adapt its behaviour depending on the circumstances /Esm-1/. This is enabled by the fact that the sequence of actions is determined at runtime.

- with some imagination, it can be said that a KBS has to search the algorithm to apply at program runtime. This search process takes time and makes a KBS potentially slower than an algorithmic program for the same task. Consequently, if an algorithm can be written it is usually not advantageous to use AI-techniques.

Conventional programs are known to be very convenient for 'number crunching': performing complex mathematical operations on large amounts of numeric data (example: finite element analysis). AI-techniques are especially useful for manipulating (lists of) non-numeric symbols in problems that require a lot of choices. Boolean variables play an important role in a KBS, while computations will be simple.

AI-techniques seem to offer advantages if a program has to explain its reasoning. The explanation facilities associated with a KBS can be very useful for the programmer during prototyping as it enables him to understand the problem better and to trace missing pieces of knowledge. In addition it makes a KBS potentially useful as teaching or training system.

9.2 AI-techniques in DIDACOE.

There was a strong desire in the research environment (the section Engineering Design of the faculty) to know more about the applicability of AI-techniques in and around CAD-systems and to gain at least some experience in working with these techniques. After a literature study on the possibilities of the use of AI in relation to CAD /Wie-6/ this desire even increased. The study showed that the use of AI-techniques could potentially be advantageous for
the interpretation of CAD-data and for process planning tasks (among others). Considering these potential topics it is not surprising that study and use of AI-techniques was integrated in the DIDACOE-project. Considering the strong desire to gain practical experience it is also not surprising that the use of AI-techniques almost became an objective, rather than one of the possible means of implementation of the DIDACOE-programs as it should have been. Yet there seemed to be some good reasons to implement at least part of the programs as a knowledge-based system:

- CAPP-literature /Sub-1,Erv-1,War-2,Sch-11/ clearly indicates that the use of AI-techniques can be advantageous during the development of generative process planning systems (4.4). Some authors almost present it as a requirement. Thus the global process planning program and the tool layout design program seemed serious candidates for implementation as a knowledge-based system. After a superficial reconnaissance of these tasks /Wie-5/ such an implementation could well be imagined,

- judging the detailed analysis of the strip layout design problem and the structure of the data involved /Wie-5/, it seemed very attractive to be able to use an object-class structure for representation of knowledge and data. The usefulness of a production rule system with search strategies could be imagined, but was less clear,

- the strip layout design program was conceived as a prototype that would have to be changed and extended several times before reaching any 'mature' state. This would be easier in a KBS,

- a complete strip layout design program that considered all influence factors would be quite complex, although it did not seem impossible to implement it as an algorithmic program. The generation of recommendations for changes in the design seemed to be easier to implement in a KBS. (This task can probably be compared with an explanation or teaching task.)

Even the interpretation program for the generation of the basic feature description and the cost estimation modules seemed possible candidates for the use of AI-techniques. Ideas for an algorithmic implementation were well under way however and trying to develop all programs by fast prototyping using AI-techniques seemed to big a risk, considering the lack of experience. At the end of the project time lacked to implement the global process planning program and the tool layout design program, thus leaving the strip layout as the only possible candidate to test the usefulness of AI-techniques.

Consequently, the decision to implement the strip layout design program as a knowledge-based system can partly be motivated by considering the specific advantages of AI-techniques and the type of problem. For another part however, the lack of time and the strong desire to 'do something with AI' more or less forced the decision.

9.3 Selection of an environment for KBS—development

A wide range of tools for the development of knowledge-based systems, from very sophisticated and flexible to very simple and limited, is available. It is hard to classify them properly as they form in fact what Harmon /Har-2/ calls a 'language–tool-continuum'. Yet, some ill defined, overlapping categories may be
distinguished that give a useful general indication (figure 9.3): languages, environments (hybrid tools), structured rule-based tools, simple rule-based tools and inductive tools. A short description of these categories follows, except for inductive tools. See /Har-2/War-3/Wie-11,12/ for details.

Languages include conventional languages like Fortran, Pascal and C, typical AI-languages like Lisp and Prolog and object-oriented languages like Smalltalk, C++ and Flavors. Lisp and Prolog are especially suitable for the development of knowledge-based systems. The main advantage of the use of languages is flexibility: the programmer can make anything he needs. At the same time this is a disadvantage. The development will be time consuming and implementation of the knowledge representation structures and search strategies requires a considerable knowledge of the underlying AI-theory.

Rule-based tools offer an inference mechanism and an editor to declare 'if...then...'-like rules. In principle, object-like structures are not available or they are very limited. The fact that search strategy and rule system are predefined implies that the user can concentrate on the description of the domain knowledge and that little or no knowledge of the AI-backgrounds is required. Potentially, programs can be developed very fast by relatively unexperienced programmers. The fixed search strategy limits the usefulness of the tools however to types of problems for which this strategy is suitable (usually diagnosis). The tools differ in possibilities offered by the rule syntax, in possibilities to influence the inference mechanism, in possibilities to structure the knowledge-base and in availability of interfaces to conventional programming languages or databases.

<table>
<thead>
<tr>
<th>LANGUAGES</th>
<th>HYBRID TOOLS</th>
<th>RULE-BASED</th>
</tr>
</thead>
<tbody>
<tr>
<td>LISP</td>
<td>ART</td>
<td>LEONARDO3</td>
</tr>
<tr>
<td></td>
<td>NEXPERT</td>
<td>LEONARDO02</td>
</tr>
<tr>
<td>PROLOG</td>
<td>KEE</td>
<td>ACQUAINT</td>
</tr>
<tr>
<td></td>
<td>GOLDWORKS</td>
<td>EXSYS</td>
</tr>
<tr>
<td>POP11</td>
<td>KNOW.CRAFT</td>
<td>PC-PLUS</td>
</tr>
<tr>
<td>APES2</td>
<td>BABYLON</td>
<td>XI-PLUS</td>
</tr>
<tr>
<td>POPLOG</td>
<td>STRATA</td>
<td>NEXT1</td>
</tr>
<tr>
<td>SMALLTALK</td>
<td>DELFI3</td>
<td>DELFI2(+)</td>
</tr>
<tr>
<td>FLAVORS</td>
<td></td>
<td>ENVISAGE</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>GURU</td>
</tr>
<tr>
<td>PASCAL</td>
<td></td>
<td>OPS5</td>
</tr>
<tr>
<td>FORTRAN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STRUCTURED</th>
<th>SIMPLE</th>
<th>INDUCTIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEONARDO1</td>
<td></td>
<td>1ST-CLASS</td>
</tr>
<tr>
<td>EXSYS</td>
<td></td>
<td>TIMM</td>
</tr>
<tr>
<td>PC-EASY</td>
<td></td>
<td>RULEMASTER</td>
</tr>
<tr>
<td>INSIGHT2</td>
<td></td>
<td>EXPERTISE</td>
</tr>
<tr>
<td>VP-EXPERT</td>
<td></td>
<td>SUPEREXPERT</td>
</tr>
<tr>
<td>EXTRAN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 9.3:** A lot of different tools for the development of knowledge-based systems exist. It is hard to classify them. The subdivision in languages, hybrid tools, structured rule-based tools, simple rule-based tools and inductive tools has been taken from Harmon /Har-2/. The tools have been placed on this 'language-tool-continuum' as 'honestly' as possible considering available information and the limitations imposed by one-dimensional classification.
Hybrid tools or development environments offer the use of objects and classes next to production rules. Facilities to define objects and their relations can differ considerably. The programmer typically has extensive means to influence the search strategies and inheritance mechanisms, to structure the knowledge-base and to interface with other programs. Some of the larger tools come close to languages in flexibility and require the associated knowledge and experience.

The following demands were formulated for the selection of a tool for the development of knowledge-based systems in and around CAD-systems in general, and more in particular for process planning tasks in DIDACOE:
- knowledge representation in rules and in an object–class system,
- availability of both forward and backward chaining,
- excellent interfaces to other programs in general and to Fortran programs in particular. This refers both to calls from conventional programs to the KBS (embedding) and to calls from the KBS to routines in conventional languages,
- a very friendly user interface that enables students to work with the environment after some days of training,
- it should not be necessary to know a lot about underlying AI-theory,
- availability on Vax under Vms and preferably on a wide range of hardware platforms,
- costs below twenty thousand dutch guilders.

The systems that were interesting from this point of view were found among the large structured rule-based tools and the simple hybrid tools. Most attention was given to: Nexpert Object, Goldworks, Delfi-3, S1, Kes-2 and Leonardo-3. The former three looked most promising. Of these, the availability of Delfi-3 on Vax under Vms was doubtful and a good user interface lacked. Goldworks' LISP-environment seemed to pose problems in interfacing it with Fortran programs, a version on Vax was announced but not available yet, and the tool seemed a bit too complex to enable quick learning and use. Consequently, Nexpert Object was selected. Its main advantages are a very friendly graphical interface, excellent interfacing facilities and good possibilities to represent objects and object relations. Some weaknesses, as compared with Goldworks, were accepted (see details in /Wie-11,12/).

9.4 Nexpert Object

Nexpert Object is a product of Neuron Data (United States) that is distributed in The Netherlands by Inter Access Systems BV. It is a hybrid tool for the development of knowledge-based systems. Some main characteristics of Nexpert will be explained below, in an addition to the general explanation of AI-techniques in paragraph 9.12.
Rules in Nexpert have the following characteristics:
- structure: If 'Conditions' Then 'Hypothesis' And 'Actions'.
- Conditions can only be combined by the 'AND'-operator. The syntax 'If (A or B) Then Z' is not allowed. Two separate rules are required instead: 'If A Then Z' and 'If B then Z'.
- Conditions will typically test the value of a boolean variable (true,false), the numeric or string value of an attribute, or the value of some mathematical expression that contains attributes. It is also possible however to check if there is 'any' object that meets specified conditions, or whether 'all' objects in a specified collection meet some specified conditions.
- Hypotheses are always boolean variables that evaluate to a value 'true' or 'false'.
- a lot of different types of Actions are possible. The most important ones are: Create or Delete Objects or Object relations, assign values to object attributes, execute an external program, perform database read/write, show a picture, load new group of rules and reset value of some attribute to unknown.
- all rules can be used in both forward and backward chaining. The program has a default procedure to decide between the two (it will switch between forward and backward reasoning automatically whenever appropriate). The same default procedure determines the order of firing of the rules (depth first, breadth first). The programmer has extensive possibilities however to change this default behaviour, both globally and locally.

The class-object system in Nexpert resembles the general system described in 9.12. Some additional remarks:
- objects can be instances of more than one class and can have more than one parent object. Classes can have more than one parent class.
- it is not possible to add information to relations between objects. Only parent-child relations can be defined. 'Part of'-relations can not be distinguished from 'is used by'- or 'is made by'-relations,
- there is a default procedure to determine which information inherits up or down and to determine in which order information is inherited. The programmer has extensive facilities to change this default behaviour,
- attributes in Nexpert are stored in a slot in an object or class. A 'metaslot can be specified for any slot. This metaslot consists of several parts, among which an 'Order of Sources' (OS) slot and an 'If Changed' slot. The latter is used to specify actions that have to be performed when the value of the attribute changes. The OS-slot specifies one or more sources that can be used by the program to derive a value for the attribute ('If Needed' slot). The most important sources are: inherit value or method to determine value from class, parent or child, assign a value using some expression, read value from database, call external routine, derive a value using the rules, use default value or ask user. The program will use the sources in the specified sequence until a value for the attribute is found. A default procedure is used for empty OS-slots.

Normally every rule can fire only once. After firing, its hypothesis is confirmed or rejected and its actions are performed, thus the rule is not interesting any
more and will be ignored by the program. Using a reset action, a hypothesis can be reset to value unknown, and the corresponding rules become available for firing again. This permits the implementation of loops over one or more rules.

Trying to derive the value of a numeric attribute, the program will never test rules, as the backward search process is limited to hypotheses: boolean variables. Rule-Actions (potential sources for value determination) are only performed if the associated hypothesis is proven true. For all other purposes, including the backward search, actions are ignored. This leads to difficulties in OS-slots if the method of value determination depends on the circumstances. This 'conditional value determination' is solved by forcing the testing of an auxiliary hypothesis from the OS-slot (by assigning its value to itself) and by writing two or more rules that have this hypothesis as conclusion. The actual value assignment is now done in the action part of the rules:

OS-slot for Object.Property:
Reset Value_Determined
Do Value_Determined Value_Determined

Rules:
If Conditions_1 Then Value_Determined And Do Value_1 Object.Property
If Conditions_2 Then Value_Determined And Do Value_2 Object.Property

9.5 Strip Layout Design as KBS

Current program limitations and the theory of strip layout design have been discussed in chapter 8. The description here is limited to the global characteristics of the implementation of the program in Nexpert Object. By lack of space it is not possible to discuss the implementation in detail. In addition this would probably not be very useful.

The strip layout design program is started by executing a Fortran program. This program performs the following tasks:
- read the basic feature description from the NXP-files (chapter 7) and store all information in Common blocks. This makes the data available to any Fortran routines that Nexpert might want to call. It avoids the need to specify transfer of these data from Nexpert to these routines at every call,
- initialize the Nexpert environment and load the first knowledge-base,
- declare available Fortran routines to Nexpert,
- volunteer the value 'true' for the variable Strip_Layout_Needed and start the Nexpert inference engine. This transfers program control to Nexpert.

The first knowledge-base (KB) in Nexpert is simple: it contains only five rules. This Start-KB can be conceived as a high-level program control. Essentially it ensures that the program performs the following actions in the presented order. Each of these actions is performed by one or more separate knowledge-bases:
- read Outer Contour data from the NXP-files. Thanks to the NXP-file-format this is easy. Using one rule, with an extensive action part, every line from a file (figure 7.7) is automatically translated into a Nexpert object with the associated attribute values.

- create Angle Groups of LIN (AGL), Line Groups of LIN (LGL) and Circle Groups of RAD (CGR) and determine their suitability. This is done by dynamic object creation from loops over groups of rules,

- create a circle-edge model and/or a straight-edge model depending on the existence and characteristics of AGL's, LGL's and CGR's,

- generate one or more layouts depending on the existing edge model and its characteristics,

- write layout data to NXP-format-files.

The program has the following global characteristics:

- A total of 367 rules is used for the following tasks: 158 for strip layout, 170 for edge creation, 29 for AGL-, LGL- and CGR-creation, 5 for the start-KB and 5 for reading OC-data and associated object creation.

- A total of 41 classes has been defined that together contain 373 slots for the storage of 152 different types of attributes. Runtime these classes will lead to a number of dynamic objects, that depends on the shape of the sheet metal part. In addition 211 fixed objects (simple variables) are used, of which 126 are boolean hypotheses.

- Approximately 120 Order-Of-Sources metaslots and 20 If-Changed metaslots have been specified. The OS-slots use the following methods for value determination: 48 times the conditional value determination principle (this accounts for the major part of the rules), 32 times value assignment using some function of other attributes, 30 times a default value and 10 times execution of an external Fortran routine.

- Sixteen different Fortran routines are called either from the action parts of rules or from OS-metaslots. Most of these routines perform simple value determination tasks. One routine is very extensive and complex: it generates an offset contour and applies the brute force angle variation technique described in 8.9.3. to determine the pitch.

- the action part of the rules is used very often to create objects, to assign values to attributes and to force evaluation of hypotheses.

### 9.6 Nexpert evaluation

The first experiences with the Nexpert environment are good. The user interface is excellent: easy to work with, clear presentation of all relevant information, and all required facilities are available at any moment. Only some error messages need improvement. The interface to Fortran routines and to NXP-format-files leaves little to be desired, although some minor problems were encountered in reading data from the NXP-files.

The knowledge representation in rules and objects is very flexible. Nevertheless it is a disadvantage that no information can be attached to relations between objects. In addition it would be helpful if an attribute could have a list of numeric constants or a list of objects as value. We now had problems in
telling Nexpert that the sequence of Outer Contour Elements (child-objects) in the Outer Contour, Edges and Edge Holes (parent-objects) is important. The facilities to influence the search strategy and the inheritance mechanism seem adequate and very flexible.

For two reasons the evaluation of the implementation of the strip layout design program in Nexpert is difficult. The first reason is that the program is not finished and that the most complex parts still have to be added. Currently the program ignores a lot of influence factors and it does not give any recommendations to adapt the design yet. The decision to use a KBS was partly taken because of this additional complexity however. Moreover Nexpert still has to prove that it allows programs to be adapted and extended easily. The second reason is that problems related to the specific implementation were mixed with general problems caused by the lack of experience. We had to get used to the new way of programming and we still had to learn how to use the Nexpert facilities efficiently. Although some tests were performed beforehand, most of the insight was obtained during the implementation itself. Consequently the conclusions below are temporary.

It took eight weeks to learn to work with Nexpert and to implement the whole current program, exclusive the associated Fortran routines (this required another eight weeks). Considering the lack of experience this is quite fast, although complete implementation in Fortran would probably have been faster, thanks to experience. The main advantage of program development in an environment as Nexpert is probably that the programmer can concentrate on describing how things should be done. Comparatively little attention has to be paid to the question when it should be done, as this is handled to a large extent by the predefined inference mechanism. In general, the available class, object and rule structures reduce programming efforts to a minimum.

It is quite remarkable that a lot of the rules essentially are of the type 'If Conditions Then Actions'. The hypothesis, that has to be present, is often of minor importance. Most rules are used in the conditional value determination which is initiated from OS-metaslots. As explained at the end of paragraph 9.4, the hypotheses of these rules are just an aid to enable firing of the rules, a kind of method reference. It should also be noted that the real rule-based search mechanism is not widely used. The evaluation of rules is often forced from the action part of other rules or from OS-metaslots. For a part this may be due to lack of experience which leads to some 'fear' to leave the order of execution undefined. It is more probable however that the problem itself clearly points to an algorithmic solution. This is especially true for the first two phases of the program: the creation of AGL's, LGL's and CGR's and the creation of the Edge Model.

Considering the experienced usefulness of the class-object system (and especially the OS-metaslots), the 'strange' use of the rule system and the limited use of the real inference mechanisms, the current program is maybe better characterized by 'object-oriented programming' than by 'knowledge-
based system. Nevertheless Nexpert proved to be quite suitable for this kind of programming.

Looking into the future it is probably better to implement the creation of AGL's, LGL's and CGR's in Fortran. The sequence of actions is completely fixed, the rules to find and define the Groups are simple, and the whole approach is not expected to change.

For the second part, Edge Model creation, this is less clear. It will probably be necessary to extend or adapt the edge model creation and to experiment with different definitions. The introduction of straight edges with 'protrusions' next to edges with edge holes, and the increased attention for the restriction of sizes of EHL's and CHL's will lead to additional complexity. Although the current part of the program could be implemented in Fortran, the implementation as a KBS may offer advantages in the future.

The third part, the actual layout creation, is probably the most suitable for implementation in a KBS. The first two program parts do little more than generate the objects that describe the sheet metal part in a convenient way. The real choices have to be made in the third part. It is here that heuristic knowledge will be required and it is also here that the rule system will have to prove its value. Unfortunately, this is not so clear from the current implementation. This is due to the facts that many influence factors have been ignored and that no recommendations to adapt the design are issued yet.

For the time being it can be concluded that Nexpert is suitable for the fast development of programs. For program delivery it is probably better to rewrite parts of the program in some conventional language, using algorithms. This will speed up program execution. Inter Access, the local supplier of Nexpert, confirms that they also use the environment in this way.
10 Global Process Planning

10.1 Introduction

Zdeblick /Zde-1/ distinguishes five types of manufacturing planning activities:
- the Facility Planning deals with long term facility requirements considering anticipated production demands over the next several years.
- the Master Planning defines the production demand that a factory must meet over the next planning period (typically one to three months).
- Production Planning considers the orderly management of facilities and the flow of material and orders through a factory. Production plans include the scheduling of labour, machinery and other resources in the production of many parts, assemblies and products. Focus is on efficient utilization of the complete factory's resources to achieve the current delivery dates or the inventory requirements.
- Process Planning (here Global Process Planning) addresses the technology of 'how to manufacture a part'. The process plan primarily specifies the raw material shape and size, required operations, the appropriate operation sequence and the workstations capable of performing these operations. It deals with general capabilities of processes and with general process economy.
- Operation Planning (here Detailed Process Planning) considers operating parameters (speeds, feeds), detailed tooling requirements, NC-programming, detailed operation times and the like. It is performed for every operation in the global process plan separately and deals with specific technology and economics of the individual operation.

Planning in DIDACOE is not production oriented but design oriented. Its objective is 'only' to test a design for manufacturability and to get the information required for a cost estimate. For detailed estimating the planning has to be almost complete, but the question 'when' a part will actually be produced is not addressed. This implies that DIDACOE can not determine the specific machines on which an object is to be manufactured, as this choice depends on optimal scheduling considering the complete production within a specified planning period. Yet the cost estimating module requires machine rates and these are most conveniently stored related to specific machines. It is assumed therefore that specific machines are selected during detailed process planning based on minimal operation cost.

Thus planning in DIDACOE primarily addresses the question 'how' a part should be manufactured and it only temporarily chooses best machines. It only considers the part currently under investigation and does not take into account other parts or products that will maybe be produced simultaneously. From this point of view only (global) process planning and operation planning are relevant.

It is not always clear where global process planning stops and where detailed planning starts (compare /Zde-1,Nau-1,Sub-1/ for example). In principle the global process plan specifies the operations and the sequence of the
operations in the process. The detailed plan specifies the actions and their sequence within an operation. It is a matter of definition however which collection of actions is considered to be an operation (is ‘milling’ an operation or are ‘rough milling’ and ‘finish milling’ operations?). Consequently it depends on the definitions of the types of operations, what is determined in the global process planning and what is left to the detailed planning. Even with well defined operations the two types of planning are highly interdependent however and it may not be possible to separate them completely.

Considering the production path concept (chapter 5) it is not sufficient when the global process plan only specifies types of operations. The plan should also define the actual task of each operation. Typically it will have to indicate the material that is to be removed. It can specify this directly by describing the features to be produced, or indirectly by defining the Less Worked Objects before and after the operation.

Separate global process planning programs will have to be developed for different manufacturing principles. Such a principle represents a collection of types of operations that are often used together in a production path or that are potential alternatives. It can be conceived to define the state space (figure 9.1): the scope of the program. Three possible examples of manufacturing principles are: sheet metal manufacture, metal removal operations (like turning, milling, drilling, grinding) and engineering plastics manufacture. More detailed principles (more limited scopes) can also be defined. Note however that the designer has to indicate the manufacturing principle and that a too detailed approach may require choices that can not be made by the designer from the point of view of the DIDACOE-philosophy.

It follows from the above that, prior to the development of a global process planning program, the manufacturing principle and the types of operations should be very well defined.

10.2 State of the art in CAPP

Computer assistance for designers is well developed, although current CAD-systems leave a lot to be desired. Computer Aided Manufacture (CAM) using NC- and CNC-programming is normal practice as well. To some extent the link between these two islands of automation can also be established: it is possible to generate tool travel paths automatically from CAD-geometry data on a part. The generation of real process plans from geometrical and technological data (tolerances, surface conditions, material properties) is still a manual task however. Therefore the development of computer aided (or automated) process planning programs (CAPP) is considered a bottleneck between CAD and CAM. Consequently CAPP is an important international research topic and a lot of literature is available. Although only a small part of this literature has been studied, some main conclusions regarding the progress in the development of CAPP-programs can be made.
10.2.1 Variant versus Generative Systems

Two main types of CAPP-systems can be distinguished: variant-type systems and generative systems. Systems of the former type are based on the storage of existing template process plans for groups of similar parts (part families). These plans have been designed before by human process planners. Usually a Group Technology (GT) code is used to describe the part families. After classification of a new part (the determination of its GT-code) the corresponding template plan is retrieved and slightly adapted by a human planner, if necessary.

Compared with the development of new plans from scratch for every new part, variant-type systems offer a lot of support. Yet CAPP-systems of this type have a number of disadvantages /Sch-1/,/Erv-2/,/Mer-2/,/Sub-1/,/Jos-1/:
- an experienced process planner is necessary to check the generated process plan,
- determination of the GT-code may be a problem, although automated GT-code generators are being developed,
- it is not possible to integrate an optimization strategy in the system,
- it is time consuming to create the system (at least 2000 to 5000 tested template plans are required), to adapt it to the specific needs of a company, and to reflect the influence of new machines and tools. The system is not very flexible,
- the Group Technology code does not describe the part in sufficient detail. Exact sizes are missing and relations between features are hard to derive. This essentially limits the possibilities of a variant-type system,
- variant-type systems store only process planning results and not the knowledge and experience that lead to these results. This may result in a loss of process planning expertise.

Due to these disadvantages variant-type systems are not considered suitable for the degree of automation of process planning required in the future Computer Integrated Manufacturing (CIM) concept.

Consequently most current research addresses generative process planning. The generative approach to process planning focuses on capturing the decision-making logic used by experienced process planners rather than simply recording previously generated process plans. It is a system that automatically synthesizes product and process information to create a suitable process plan for a new part /Sub-1/.

Environments like DIDACOE clearly require the generative approach.

10.2.2 Use of AI-techniques

The use of AI-techniques for the development of generative CAPP-systems is very common /Erv-2/, /Sub-1/, /Sch-11/, /Nau-1/, /Wan-1/, /Wri-1/, /Iwa-1/, /Inu-1/, /Phi-1/, /War-2/, /Mer-2/. The following reasons for the use of AI in process planning tasks are mentioned:
- no adequate models are available for the description of all relations between workpiece / material, tools, machines and functional design demands. Although limited models exist for partial problems like the determination of forces, speeds, feeds and tool materials, process planning is primarily based on empirical relations and on the heuristic knowledge and experience of experts. Production rules are a convenient means to declare this type of knowledge,

- a process planner thinks of a part as consisting of features and relations between features. Object-class systems (frames) are very useful for the storage of such descriptive schemes,

- process planning is iterative by nature. The planner explores a number of possibilities before arriving at a final solution. Planning can conveniently be conceived as a search process that includes backtracking. Consequently, AI search strategies are useful,

- process planning decision knowledge is subject to change. It should be easily possible to add knowledge on for instance new machines and new tools. Thus there is a need for ease of extension,

- the program should be able to explain its line of reasoning to serve as a training tool for inexperienced process planners (and designers).

In general it can be stated that the knowledge representation facilities (both rules and frames) and search strategies offered by AI are very convenient while developing programs for the very complex, ill understood process planning task.

10.2.3 Progress in problem solution

Progress in the development of generative CAPP-systems is still slow. According to Subramanyam /Sub-1/ this is partly due to a lack of proper understanding of the relevance of AI-techniques in the engineering domain, but the main reason is the overwhelming complexity of the task. Wang /Wan-1/ notes that most attempts have been limited to the selection of machining operations. According to him, operation sequence, tooling requirements and machining parameter selection have largely been ignored. Thus, most current programs are very limited, and all generative-type programs seem to be prototypes. It is expected that it will take another five to ten years before generative process planning systems are available for practical use /Erv-2,Sch-11/.

Most current programs deal with either prismatic or rotational parts that are to be produced by metal removal operations like turning, milling, drilling, reaming and grinding. Programs for rotational parts typically use 2D CAD-data as input. Programs for prismatic parts usually start from 3D Boundary Representation models. Apart from the completely machined part, the unworked blank is usually also defined in the CAD-system.

A typical first step in a CAPP-program is the recognition of features that have to be machined. Although it is possible to recognize some features in for instance a BREP-model, it is still difficult to write general error-free feature recognition programs. Consequently, feature recognition can be conceived as a serious bottleneck. This topic has been discussed briefly in paragraph 4.5. The
references supplied there give a general impression of the problems and possibilities.
Typically feature recognition will involve the following steps:
- define the features to be recognized,
- define how these features are represented in the CAD-model,
- write a recognition 'algorithm' for each type of feature,
- search the CAD-model of the completely worked part for features. After a feature has been found, determine its parameters and 'remove' it from the model. Repeat this recognition procedure until the part model equals the model of the unworked blank.
This provides a list of features to be machined.

The next step is to find the operations and sequence required for the manufacture of each individual feature. This is typically performed, in a reasoning process, by starting from the completely machined feature and by adding material (applying inverse operations) until the unworked feature is obtained. Often, more than one operation will be required to produce a feature, implying that a path consisting of several operations and less worked features is generated. In any state it will usually be possible to apply several (inverse) operations, implying that the search space consists of a tree of possible paths. In AI-terms this requires a forward chaining search process (similar to figure 9.1). Simple functions to estimate the costs of potential operations /Erv-2/Wri-1/ or heuristic planning knowledge, can be used to guide the search for the best path(s).
This supplies a list of operations and sequence per feature.

As shown by van 't Erve /Erv-2/ several cutting tools may be available to perform an operation and the same tool may be suitable for several operations. A third step in his XPLANE-program is therefore to select cutting tools per operation in such a way that the total number of tools required is as small as possible.
A fourth and last step is the determination of the overall sequence of operations for the machining of the complete part. In this phase minimal number of tool changes, minimal tool travel time and minimal number of setups may act as guides. The sequence of operations per feature is a constraint. In addition it may be necessary to machine some features before others.

Further study would be required to give a full description of the 'state of the art' in CAPP.

10.3 Elements in sheet metal manufacture

10.3.1 Types of Basic Objects

Sheet metal manufacture will typically start from one of five types of Basic Objects: Wide Coiled Strip, Coiled Strip, Strip, Sheet or Blank. These objects are characterized by a constant thickness (within tolerances) that is very small as compared to the other dimensions.
A Sheet is a rectangular shape with typical dimensions ranging from 1000*2000 to 1500*3000 mm. A Strip is a rectangular shape as well, but its length to width ratio is much larger than that of a sheet. The width is associated in some way to the dimensions of the parts to be made, as defined by the strip layout, and maximally circa 400 mm. Lengths range from 700 to 3000 mm. This limited length distinguishes them from Coiled Strips, that can be several dozens of metres long. Wide Coiled Strip is the base product made by the sheet metal producer. In general its width is not related to the parts to be made, and typically 700 to 2000 mm. A Blank is a piece of sheet metal, whose shape and dimensions are related to the part to be produced. The shape will usually be simple, typically rectangular or circular. (Note that the term 'Blank' is also used more in general to indicate intermediate objects).

Manufacturers of sheet metal parts will buy Wide Coiled Strip if the quantity of material needed is sufficiently large. They will then produce the other shapes themselves. An alternative is to buy Sheet or Coiled Strip. In a lot of cases, Strip and Blank will not be bought: the part manufacturer will often produce them himself, typically by shearing from Sheet. When buying, not all basic object dimensions are standard available. Only specific thickness-width-length combinations for specific material types and qualities are available 'of the shelf'. If the required quantity is sufficiently large however, any dimensions can be obtained, but costs may be higher (depending on quantity) and delivery time may be longer. From a process planning point of view this implies that a 'make or buy' decision has to be made, especially for Strip and Blank.

Basic sheet metal objects can be bought bare or coated. In the latter case the material supplier has already applied some surface layer, using painting, electroplating, oxide coating or other processes. To avoid damaging of this surface layer during part manufacture, special protection layers (that can 'easily' be removed later) can be obtained as well.

For the process planner this implies a choice between the purchase of coated material and the purchase of bare material with the need for finishing operations.

10.3.2 Types of Operations

The number of types of operations applied in sheet metal manufacture is very large /Din-1,Nni-1,2/ It is not attempted here to give a full survey or classification: only some of the most important types of operations will be mentioned. As shown in figure 10.1, four main types can be distinguished: cutting operations, bending or forming operations, operations for complex deformations and finishing operations.

The finishing operations include surface treatments (painting, electroplating, oxide coating), heat treatments, deburring, degreasing, polishing, tapping, drilling small holes and the like.

Operations for complex deformation include all kinds of (deep)drawing, forcing and pressing. They fall beyond the scope of this thesis.
Bending or forming operations produce straight bends, as indicated in paragraph 4.8. Some important variants are figure 10.2:
- swing bending, where the workpiece, blank or sheet is clamped between the machine table and a pressure beam while a bending beam forces the material to bend by rotating around an imaginary straight line,
- 'slide' bending, which resembles swing bending, except that the bending beam performs a translation instead of a rotation,
- die bending, where the workpiece is formed between two die parts. The surface of both of the die parts corresponds with the shape to be obtained,
- 'saddle' bending, which is the same as die bending except that only one of the die halves has a shape that corresponds with the bend to be produced. The other die half merely supplies the required pressure,
- free bending, where none of the two die halves has the shape of the bend to be produced. It is a three-point bending process.

Cutting operations separate material. They can be divided in two main groups: mechanical and non-mechanical operations. Operations of the latter type do not use a solid tool but some kind of medium. Examples are oxygen cutting, laser cutting, plasma cutting and waterbeam cutting.
Mechanical operations use solid tools that consist of two parts. One of the parts moves, while the other is fixed or moves in opposite direction. There is a wide variety of types of operations /Nni-1/, with two main variants: shearing and punching. Although the difference between them is sometimes disputable, main characteristics are indicated below.
Shearing uses two blades that make a small angle with each other (like a pair of scissors) and that progressively cut the material during a stroke of the machine. Five main subvariants can be distinguished (figure 10.3):
- slit cutting: continuous cutting along a straight line, for example by using rotary shears,
- straight shearing: cutting along a straight line with limited length, for example by using guillotine shears,
- contour shearing: cutting along an arbitrary contour with limited curvature using serpentine or circular shears,
- trimming: cutting irregular edges from a workpiece using circular shears,
- shear-nibbling (dutch: 'knibbelen'): cutting along an irregular contour by repeated action of very short straight shears (vibratory shears).

Punching uses a punch and a cutting plate as illustrated in figure 6.1. The main difference with shearing is that the shape of the cut line is already present in the tool. The contour to be cut is often closed in itself and cut lines with high curvature can be made accurately. In addition it is usual to cut the material everywhere at the same time, and not progressively as in shearing (edges of punch and cutting plate are in parallel planes). Some major subvariants are:
- blanking: cutting around the periphery of a part to separate a 'blank' from the surrounding material,
- piercing: cutting holes in a workpiece,
- cut-off: separating a workpiece from a larger piece of sheet metal (see chapter 6),
- notching: removing corners or parts of edges from a workpiece,
- punch-nibbling (dutch: 'knabbelingen'): repeated punching along an arbitrary (usually closed) line using a small standard punch.
Figure 10.3: Three types of shearing operations. (a) slitting using rotary shears, (b) straight shearing using guillotine shears, (c) shear-nibbling using vibratory shears.

10.3.3 Types of Machines

The number of types of machines almost corresponds with the number of types of operations. A lot of examples can be found in /Nni-2,3,4,Tim-1,Ost-1/. Some important types are:

- general purpose presses, used for punching, bending and drawing. Main variants are mechanical and hydraulic presses and single, double and triple action presses /Nni-4/.
- power (guillotine) shear machines for straight shearing,
- all kinds of special shearing machines /Nni-3/,
- turret punch press machines that use a variety of standard punches for repeated piercing or notching of blanks or sheets /Tim-1/,
- power press brake machines, especially for die bending, saddle bending and free bending,
- bending brake machines for swing bending,
- all kinds of special bending machines /Nni-2/,
- laser cutting machines.

Next to these machines that actually do the work, a lot of auxiliary equipment is available such as coil winders, strip straighteners and automatic feeding and removal equipment. Recently a lot of attention has been given to lowering preparative and secondary manufacturing times. Flexible Manufacturing Systems have been introduced for 'just in time' manufacture of small batches of sheet metal parts. These systems show a high degree of automation where a computer controls the actual operations, the transport and change of tools and the transport and handling of raw materials and (un)finished parts.

Next to the automatic part and tool handling systems especially turret punch press machines are being extended for more flexible use. Machines that combine turret punching with laser or plasma cutting are already available. The addition of deburring, drilling and tapping facilities is expected /Tim-1, Fel-1, Bar-1/.
10.3.4 Types of Tools

Solid tools for punching and forming can be divided into two major groups: standard tools and special tools.

The characteristic of standard tools is that they have standardized shapes and dimensions. They are comparatively small, typically have simple shapes and are not intended for the production of one type of part only (they are multi-purpose or useful for families of parts at least). Some examples are shown in Figure 10.4. Standard tools can be used separately, combined with other standard tools in adjustable master die sets/Bra-1/ or as part of special tools. Of special importance is their use on turret punch press machines. These machines may have as much as hundred different standard tools immediately available in their local tool storage, allowing a wide variety of parts to be made. Standard tools will typically be used when production quantities are low.

Special tools are intended for the production of one type of part only. These tools are part specific instead of shape and dimension specific. Special tools are used on multi-purpose presses for large quantity production. Two major types can be distinguished: progressive dies (see chapter 6) and conventional dies. Tools of the latter type contain a limited number of punching and/or forming tools that have been assembled together in one die set. A conventional die can be compared with one step of a progressive die, but now each die is fixed on a separate press.

Compound dies, fine-stamping dies and finish-stamping dies (shaving dies) are special tools that are used to manufacture high-accuracy parts/Wie-5/.

Figure 10.4: A selection of standard punching tools (Source:/Boe-1/)
Compound dies /Her-2, Gei-1, Rom-1, Bla-1/ typically combine a blanking operation with one or more piercing operations. All operations are performed on the same area of sheet metal in one stroke of the press. The blanking punch also serves as cutting plate: it contains one or more openings that act as cutting holes for the piercing operations. Compound dies will typically be used to meet small distance tolerances between holes and the outer contour, to meet flatness tolerances or to get all burrs on the same side of the part. They can be used separately (as a conventional die) or as the last step in a progressive die.

Finish-stamping or shaving dies /Her-2, Gei-1/ are used to obtain high dimensional accuracy and smooth cut faces with sharp edges. Finish-stamping is applied after a piercing or blanking operation. The latter operation essentially shapes the hole or outer contour but leaves a small amount of material (the allowance, 8 to 10 percent of sheet thickness) to be removed by the shaving tool. Shaving dies can be used separately, like conventional dies. In this case it is also possible to use special vibrating presses. Alternatively they can be used as a tool element in a progressive die. (Shaving dies remove small pieces of material that can easily damage the die. Due to this risk, they are not applied very often any more).

Fine-stamping is cutting with a very small clearance, using a low cutting speed and complete fixation of the material. V-shaped grooves are pressed into the (waste) material to fix it, and to move additional material towards the cut line. Additional fixing of the material is obtained by counterforces that work on the areas that are to be cut out. Fine-stamping dies are used for the same reasons as finish-stamping dies. The additional accuracy is now obtained directly during punching however and not in two or more operations. Fine-stamping requires a special triple action press /Her-2, Gei-1/.

10.4 Possibilities for CAPP in sheet metal manufacture

Until now most attention in the DIDACOE-project was given to cost estimation and to the detailed planning for the production from strip in progressive dies. Apart from a general reconnaissance /Wie-5/ global process planning has been ignored. Consequently the description of the elements in the previous paragraph is too general and too limited to be useful. An extensive study will be necessary to:
- define the manufacturing principle 'sheet metal manufacture': the scope of the global process planning program,
- define the types of operations,
- define relations between types of operations and types of machines and tools,
- quantify the capabilities of operations, machines and tools, for example by specifying obtainable accuracy, maximum forces, maximum part dimensions, minimum and maximum sheet thicknesses and the like,
- find heuristic information that a process planner uses to decide between alternatives.
Although such a study has not been conducted yet, a number of remarks regarding global process planning for sheet metal manufacture can be made:

- it seems virtually impossible to create a process planning program for a manufacturing principle that includes all sheet metal operations. The scope of the program will have to be limited to, for instance, all the operations currently used in a specific company (this will also offer advantages when trying to collect heuristic planning information or quantified data on operation, machine or tool capabilities). Alternatively a selection of the most common types of operations can be used.

- although simple at first sight, the definition of operations will pose considerable problems. This is due to the fact that sheet metal processing is often not characterized by clearly separated, sequentially performed operations with well defined intermediate less worked objects. Usually it will be necessary to consider a collection of sequentially or simultaneously performed partial operations as one operation in the sense of the production path concept (chapter 5). It would hardly be feasible for instance to consider every piercing operation on a turret punch press as a separate operation. This is also the reason that 'production from strip in a progressive die' is conceived as one operation.

This raises the question what operations or collection of operations should be conceived as operations in the sense of the production path concept. A general answer to this question is hard to give. To some extent the answer will depend on the usefulness of an operation as a unit during detailed planning or cost estimation. In addition it is probably better not to distinguish two operations in the production path if it is difficult or 'artificial' to define a less worked object as intermediate state.

- process planning will start by recognizing the features that have to be produced. Next, one or more partial operations per feature will have to be identified. After this it will be important to know how and when partial operations can be combined. Different types of combinations can be distinguished:

1. in the same partial tool. Examples: piercing and shaving, or cutting free and bending combined in one punch,
2. not in the same partial tool but acting simultaneously on the same area of sheet metal that is associated with one eventual part. Example: all partial operations in a conventional die or in a step of a progressive die. Blanking and piercing can not be combined in this way, except in special cases as compound or fine-stamping tools.
3. acting simultaneously on different areas of the same input object. For instance: partial operations in different steps of a progressive die (see 8.11),
4. acting sequentially on the same machine and same input object. For instance: all partial operations on a turret punch press. Notching and piercing can be combined, but bending possibilities are limited on such a machine.

It will not be easy to define all admissible combinations.
it is more difficult to define the raw material shape and size for sheet metal parts than for machined parts. First, different types of basic objects are available (coil, strip, blank, sheet). The type has to be chosen before dimensions can be determined. Second, it is common practice to produce more than one part from the same basic object. This implies that basic object dimensions are not directly related to part dimensions: sheet and strip layout problems are introduced. Third, there is a choice between coated and bare material.

the second remark in the previous point introduces two additional problems. First, feature recognition is complicated because it may be necessary to distinguish additional (non-part) features between adjacent blanks in for instance a strip layout. Second, it is possible to produce different types of parts or less worked objects from the same basic object (note that this has been excluded in DIDACOE!)

- it is probable that a sheet metal manufacture process will be imagined to consist of three subprocesses: preparative operations, main sheet metal operations and finishing operations. It also seems likely that it will be advantageous to distinguish a limited number of different types of main sheet metal processes. Four examples are shown in figure 10.5. The choice between these types of processes will probably be made first, depending on global information such as quantities to produce, required accuracy, general part dimensions and sheet thickness. Such a choice clearly indicates types of tools and types of machines or considerably limits the search process at least. It more or less defines the required finishing operations (everything that can not be done in the main process). The input objects for these main sheet metal subprocesses can be conceived as semi-basic objects: they can either be made from real basic objects using preparative operations or they can be bought directly.

Summarizing, it can be stated that process planning for sheet metal manufacture differs considerably from planning for the machining of rotational or prismatic parts and that it will probably not be easier. Nevertheless it will be useful to study the use of AI-techniques in existing CAPP-programs and the application of the 'inverse' reasoning mechanism in more detail. Considering the slow progress in CAPP for machining, and the remarks above, the development of a generative process planning program for sheet metal manufacture will be very difficult.

10.5 Generation of manufacturability information

Subramanyak /Sub-1/ distinguishes a 'part understanding stage' as the first activity of a process planner. In this stage the process planner tries to get a general impression of the manufacturing requirements for a part, and he will decide here whether to continue the planning process or to request the design department for suitable design changes. That is: a human process planner will give manufacturability information to the designer.
Figure 10.5: It is probably convenient to distinguish three subprocesses in a sheet metal manufacturing process (top): preparative operations, main sheet metal operations and finishing operations. Main sheet metal operations will probably be selected first during process planning, based on global part information. Some types of main operations and their (semi-)basic objects are indicated (bottom).

Existing CAPP-programs usually ignore this feedback of information (see remarks in paragraph 3.12 however). This is not surprising: the development of CAPP-programs already poses severe problems when the design is conceived as a given fact, without worrying about possible changes. Yet it is essential for an environment like DIDACOE that manufacturability information can be generated. This type of qualitative information is maybe even more important than the quantitative cost information, as it is easier to relate it to features in the design.

Some general remarks can be made considering the generation of manufacturability information during process planning:

- in general, almost anything can be made. In practice however the possibilities are often limited by the available production facilities and by the experience in working with these facilities. Consequently it will be possible to define some features or combinations of features that can not be made given the circumstances.
In most cases however the question will not be whether a design can or cannot be manufactured. Usually it will be necessary to judge which solution is easier (more economic) or more difficult (more expensive). In principle this implies that a process planning program has a choice: accept the design and produce a plan or comment on the design or both.

- it should be possible to develop programs that can deduce themselves, by reasoning with basic information, that some features or feature combinations are difficult to make. Such programs could discover design rules and they could explain them by showing the reasoning process that lead to their discovery. For the time being this will be very difficult however. As most design rules are already well known (see /Bra-1, Bod-1, Bre-1 for sheet metal), it is much more practical to feed the program with a list of undesirable features and feature combinations. It can then check if features from this list appear in the feature description of the part. If so, the program will alert the designer and present the predefined explanation.

- some design rules may be generally valid within the context of a manufacturing principle. For sheet metal manufacture however it makes a lot of difference whether special tools are used, whether standard tools are applied or whether non-mechanical cutting is selected. This implies that possibilities to comment on a design depend on the amount of detail of the process plan. Manufacturability information will probably be supplied at different levels of detail during process planning. This may make it very difficult to implement a general 'part understanding stage', as suggested by Subramanyam /Sub-1/.

- the generation of manufacturability information during process planning may introduce a fundamental difficulty. Typically, a program will first choose specified types of operations based on part features. After a type of operation has been selected, it can check the associated list of undesirable features, to check if design rules have been violated. If these undesirable features are present however, it may well be that the type of operation is never chosen in the first place, because the presence of these features made the program decide to select another type of operation. Preferably therefore, the decision to choose a type of operation should be based on other information than on the presence or absence of features that appear in the 'checklist' for this type of operation. If this is not possible, it will be necessary to implement mechanisms that allow the program to choose types of operations for which not all conditions are fulfilled. This may require probability reasoning.
11 Data Storage

11.1 Runtime, temporary and permanent storage

The current DIDACOE consists of five separate programs: the standard Medusa Sheet Metal Design Program, an Interpretation Program, a Strip Layout Generation Program, a Strip Layout Presentation Program and a Cost Estimation and Presentation Program. In a typical DIDACOE session all these programs will be executed in the presented order.

The programs use and provide data that have to be stored. Three types of storage can be distinguished:
- runtime storage, during program execution,
- temporary storage between programs during one session,
- permanent storage between sessions.

In Fortran programs data are stored runtime in variables and arrays. Although the data are grouped into Common Blocks, they are not really structured: Fortran does not offer datastructures like records or objects. A major part of the Strip Layout Generation Program is implemented in Nexpert Object. In this environment data are stored runtime in objects and attributes: a structured storage (chapter 9).

The original intention was to use the Medusa Relational Database (MDB) for central permanent storage of all data between programs and between sessions. For several reasons this idea was abandoned however:
- due to errors in the Medusa software it proved difficult to link both DARS-routines (paragraph 7.3.3) and MDB-Access-routines to the Interpretation Program at the same,
- MDB-Access-routines are not very friendly for the programmer,
- data storage and retrieval in MDB turned out to be rather slow,
- Nexpert would not be able to read data directly from MDB.

At the time, an alternative database management system was not available. Therefore collections of files are used for temporary and permanent data storage (figure 11.1). Four collection of files can be distinguished: Medusa-files, NXP-files, CDF-files and DAT-files.

The Medusa-files are permanent storage and contain data on 2D drawings in a Medusa internal format. Three relevant types can be distinguished:
- 'partname'.COR-files that contain the descriptions of the unfolded sheet metal parts. These files are produced by the Sheet Metal Design System and serve as input for the Interpretation Program,
- 'partname'.DID-files. These are the adapted .COR-files produced by the Interpretation Program (chapter 7). They will be used in future for the presentation of qualitative cost information in the part drawing,
- 'partname'.SLx-files (x is a sequence number) that contain the layout drawings produced by the Strip Layout Presentation Program.
Figure 11.1: Survey of data storage in the current DIDACOR-implementation. The COR-, DID- and SLx-files contain descriptions of 2D Medusa drawings. The NXP-files are ASCII-files that use the Nexpert Database Format. They temporary store the basic feature description and layout-information. The CDF-files (Cost Data Files) are direct access formatted (fortran) files. They permanently store information on Main Objects and product structure, on operations and production paths and detailed data on specific sheet metal operations, including the strip layouts. Advanced Feature Descriptions and Outer Contour Offsetline data are only stored runtime.
NXP-files are ASCII-files that use the NXP-database-format. This format is illustrated in figure 7.7 and defined in detail in /Neu-1/. The Basic Feature Description for sheet metal parts (output of the Interpretation Program) and the Layouts (output of the Strip Layout Generation Program) are stored in these files. The NXP-files can contain data on one part only: existing data are overwritten as soon as new data (for the same or for another part) become available. Consequently, NXP-files represent a temporary storage of data between programs, during a session.

An advantage of NXP-files is that they can be read directly by Nexpert programs: no additional interface is required. The fact that they are ASCII-files allows direct inspection and editing, which is convenient during program development. The fact that data on only one part are stored simplifies the references between the files and speeds up read and write.

CDF-files (Cost Data Files) are formatted direct access (Fortran) files. They permanently store all general cost estimation data (chapter 5), the operation specific estimation data (chapter 6) and the layout information. Basic Feature data are not stored in these files. CDF-files store the results of cost estimations for many different parts and products. These data can be inspected and changed interactively by using the Cost Estimation and Presentation Program separately (without the other DIDACOE-programs).

DAT-files are ASCII-files for the permanent storage of part independent ‘common’ data (see next paragraph).

Currently, the advanced feature description (the edge model) and the outer contour offset line that are derived during strip layout generation, are not saved in any way. These data are only present at program runtime.

11.2 Selection of data for storage

Data can be subdivided into part specific data and common data. The latter include material properties, machine data, interest rates, operator rates, indirect material factors, basic time data and all kinds of default values. These data are not part specific. They are stored independent of the parts (in the DAT-files) as fixed basic data and they can be conceived to belong to the program. The current collection of common data is limited to the minimum required to test the programs. They will not be described.

Part specific data are dynamic: they depend on the part that is currently being examined by the program. These data can be subdivided further into basic part data and non-basic part data. The characteristic of the latter is that they can be derived from the former. On a detailed level, the radius and length of a RAD-segment are non-basic data because they can be derived from the coordinates of the centre, start and end point of the segment. On a global level it could be stated that only the original part description in the ‘partname’.COR-file is basic information. As long as all programs execute completely automatically, all other data can be derived again by rerunning these programs.
It is not efficient to store all possible data. This implies a selection problem: which data should be stored? At program runtime it is usually convenient to define one variable or attribute for every datum that is used on several occasions, for ease of programming, readability of the programs and to avoid recalculation every time that the datum is needed. It is also obvious that all common data and all basic part specific data should be stored permanently. The main question is therefore, which non-basic data should be stored permanently.

Until now this question did not receive a lot of attention. For ease and speed of program development, DIDACOE now works with an almost one-to-one mapping between runtime and permanent stored data. As a consequence, the current temporary and permanent storage contains too many non-basic data. For a prototype program this is no problem, but a critical re-examination is required before a final program version is developed.

In general the decision whether non-basic data should be stored or not depends on a choice between the occupation of storage space and the need for recalculation time. This will not be discussed here any further. For cost information systems there are two additional aspects to consider however.

A first problem is that many cost estimation data can be either basic or non-basic, depending on the circumstances. This is a consequence of the facts that estimates can be performed at different levels of detail and that an estimation program will have different levels of knowledge on different types of operations and parts.

An example are operation times. For a general unspecified operation, the cost estimating program will prompt the user for these data. For a specific (known) type of operation the program will derive the times itself from other data. Consequently the operation times are basic data in the first case and non-basic data in the second case. This implies that it should be possible to store them.

Time dependency of common data is another factor to consider. Of course, it introduces a maintenance problem for the common data files: they have to be kept up-to-date. As can be seen from the following example however, this is not the only problem.

Suppose that costs for a part are estimated at time T using a material price P1. At time T+1 the common data files are changed to bring them up-to-date, and the material price changes to P2. At time T+2 the part estimate data are retrieved. The result of this retrieval will now depend on the data that were saved at time T.

If only basic data, such as the material identification, were stored, the retrieval will involve a recalculation of for instance the material costs. The material identification will be used as a keyword to access the common material data. As a consequence the recalculation will use the actual price P2 and not the original price P1. The estimate is updated automatically during retrieval. Due to lack of data (price P1 is not stored any longer), the original estimate is lost.

If material costs (non-basic data) were stored directly, the retrieval will not trigger recalculation, but the retrieved data will not be consistent with the new
price P2. If the existing estimate is changed however, the associated recalculation will not only take into account the change, but the new price P2 as well.

Even more nasty problems can arise when retrieved estimates reference materials or machines that are no longer present in the common data files.

For the time being this problem has been handled by storing used common data as part specific basic data. A storage facility for these data has to be present anyway, to allow the user to define new materials and machines at runtime, without the need to change the common data files. Thus, both used common data and actual common data are available at retrieval time. If there is a difference between the two, the program can alert the user.

11.3 Data structures

To organize the large amounts of data in environments like DIDACOE it is convenient to distinguish groups of related data and to treat these groups as units. In AI-terminology these units can be conceived as Objects (see paragraph 9.1.2), in database terminology they are Records. Different types of units can be distinguished. In AI-terminology these types are Classes, in database environments they will be called Record Types or Tables.

Appendices 1 and 2 describe most of the types that are currently used in the DIDACOE implementation. Each type has a name and specifies the fields in the records (the attributes in the objects). These types are used both for runtime and permanent storage. Runtime, a type corresponds with one or two Common Blocks of Fortran variables and arrays, or with one Class in Nexpert. In temporary and permanent storage, each type represents a separate NXP- or CDF-file.

As explained above, comparatively little attention was paid to the exact fields that should be present in the types at the different occasions. In contrast, a lot of time was spent in defining which types should exist (regardless of their exact contents). The main objective was to create a flexible, easily extendable hierarchy of types and subtypes that would allow for storage of different types of data depending on the type of object, the type of operation and the type of estimate.

This flexible environment was obtained by considering the following principle. Imagine a type T with a field ID (for Identification) and a field F. Suppose that it is known that the value for F will in practice be determined either:
- directly, for instance by prompting the user (method 1),
- by multiplication of A and B (method 2),
- by evaluating C * D + sqrt ( E ) (method 3), or
- by some other still unspecified method (method 4).
In cost estimation environments such conditional value determinations, and the associated data storage problems, are quite common.
Three approaches to the storage of data are now possible (figure 11.2):

1- Store all data in the same table: add A, B, C, D and E (all data that may be required to derive a value for F) as fields to type T. This procedure will result in very extensive type descriptions, while only fields associated with one of the methods will actually be used in any instance of the type. Other fields have to be added to type T in future when new methods to derive F are being specified.

<table>
<thead>
<tr>
<th>ID</th>
<th>F</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj1</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj2</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj3</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj4</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(2) Type T

<table>
<thead>
<tr>
<th>ID</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj1</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Type U

<table>
<thead>
<tr>
<th>ID</th>
<th>(F)</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj2</td>
<td>(x)</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Type V

<table>
<thead>
<tr>
<th>ID</th>
<th>(F)</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj3</td>
<td>(x)</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Obj4</td>
<td>(x)</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(3a) Type T

<table>
<thead>
<tr>
<th>ID</th>
<th>F</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj1</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Obj2</td>
<td>x</td>
<td>U</td>
</tr>
<tr>
<td>Obj3</td>
<td>x</td>
<td>V</td>
</tr>
<tr>
<td>Obj4</td>
<td>x</td>
<td>V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Type U

<table>
<thead>
<tr>
<th>ID</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj2</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Type V

<table>
<thead>
<tr>
<th>ID</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj3</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Obj4</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(3b) Type T

<table>
<thead>
<tr>
<th>ID</th>
<th>F</th>
<th>S</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj1</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obj2</td>
<td>x</td>
<td>U</td>
<td>4</td>
</tr>
<tr>
<td>Obj3</td>
<td>x</td>
<td>V</td>
<td>20</td>
</tr>
<tr>
<td>Obj4</td>
<td>x</td>
<td>V</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Type U

<table>
<thead>
<tr>
<th>ID</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Type V

<table>
<thead>
<tr>
<th>ID</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 11.2: Three different approaches to the storage of data using types and subtypes. See the text for explanation. An 'x' indicates a used field, a '-' indicates an unused field.
2- Use separate tables, depending on the method that is used to determine a value for F. Keep the existing type T with field F (for method 1), add a new type U with fields ID, A and B (for method 2) and a new type V with fields ID, C, D and E (for method 3). A field F could be added to types U and V, but this is not strictly necessary, as F is a non-basic variable in these cases. New types can now be defined for each new method. Note that the meaning of type T has changed. It does no longer store all objects of the original type T, but only those for which the value of field F is determined using method 1. As a consequence, there is no central table any more that stores all objects of the original type T. Looking for data of an object with a specified ID, we have to search all tables (T, U and V). Apparently, types T, U and V have no relation.

3- Use type T as the central type that stores all objects, and create subtypes that store continuation data depending on the method used to derive a value for F. Define a new type U with fields ID, A and B and a new type V with fields ID, C, D and E. Add a field S (for subtype) to type T. A value in field S now indicates the table (none, U or V) in which continuation data are stored. Specific records in this table can be found by searching for the object ID. Other subtypes (tables) can easily be added later without changing any of the existing types. Moreover, all fields in all records will now be used. Of course, the same principle can be repeated in the subtypes, to create a hierarchy of several levels for the description of an object or operation.

In general, the first two solutions are not recommended. Nevertheless the first approach has been used on some occasions (see the BO-, AMO- and BP-tables in appendix 1 for example).

In most cases the last solution was applied. This explains that a lot of tables presented in the appendices have a field 'subtype'. In the current implementation object identifications are only stored in the central table (the top type). Specific continuation records in subtypes are referred to by their record numbers instead of by an ID. Hence, an additional field 'subtype record number' can be found in many of the tables in the appendices.

In the runtime storage in Nexpert, the 'subtype' and 'subtype record number' fields are not necessary. They are replaced by parent-child relations between the classes. The lowest class in the hierarchy automatically inherits all fields (attributes) of the higher classes, and objects are instanced from this lowest class (chapter 9).

The above explains how one operation, one part or one other data unit can be described by several records in tables on different levels.

In addition it is necessary to define relations between different units. For this purpose, many tables described in the appendix contain a field 'X-reference', where X is replaced by 'Machine', 'Material', 'Shape' or some other type of unit. A value 0 in such a field indicates that there is no relation with a unit of type X (for instance: no shape defined for this part). Another value represents the record number of the related unit in the X-table (for instance: record number 3 in the Shape-table).
Alternatively, the relation between two units may be specified in a separate relation table. This table will have at least two fields that contain the identifications of the related units. An advantage is that other data that describe the relation can be added (compare the MOR-table in appendix 1).

More specific notes on the data structures can be found in the appendices.
12 Conclusions and Recommendations

1
Potentially designers have a large influence on eventual product costs. Due to lack of time, communication problems and lack of design oriented cost information however, they are often not able to optimize designs from a point of view of manufacturing costs. This problem arises especially where production quantities are low.
The situation can be improved by the development of special cost information tools and by appropriate organizational measures. This research considered the development of computerized cost information tools for designers.

2
A designer needs different types of cost information during the design process. Two main types can be distinguished: qualitative and quantitative information. Qualitative cost information enables the designer to find main cost influencing factors, to find useful existing solutions or to choose the best (cheapest) solution from two or more alternatives. This type of information potentially leads to cheaper products, but it does not specify costs or savings. Quantitative cost information tells the designer how much a design will cost or how much can be saved by specified changes. It is used to check if target costs will be met and to choose between alternatives.
Both types of information can be required for objects with different levels of complexity (products, assemblies, parts, features, functional units) and for different design states of these objects (conceptual, materialized, detailed).

3
Cost information for the designer should be design oriented. Primarily this means that:
- the information should be related to decisions that the designer has to make and to aspects of the design that can be influenced by the designer,
- the information should not only specify costs, but indicate as well why a design is cheap or expensive, or how it can be adapted to reduce costs.
Information that is present in process planning and cost estimating departments does usually not meet these demands. This implies that information for the designer has to be specially prepared.

4
Use of a cost information tool should interrupt the design process and disturb the designer as little as possible. This means that:
- use of the tool should not require detailed process planning or cost estimating knowledge. This limits the information for which the tool can prompt the designer,
- the tool should respond on a level that corresponds with the current state of the design process. It should not prompt for detailed information during conceptual design for example,
the tool should be linked to a Computer-Aided Design system for ease of access by the designer and to enable exchange of data whenever necessary. Double input of design data should be avoided.  
the tool should be easy to use (friendly user interface) and respond fast.

5
To support the designer during all phases of the design process, a cost information tool should include:
- database search for data on existing designs and standard parts and for qualitative cost information such as cost structures, relative costs, design rules and other manufacturability information,
- active generation of manufacturability information,
- global cost estimation methods,
- detailed cost estimation methods,
- methods for technical/economical assessment of conceptual designs.

6
Characteristic for global cost estimation is that costs for a new object are estimated by comparing it with existing objects with known costs. Cost information is prepared per class of similar objects, often in the form of some cost function.
The main potential advantages of global cost estimation methods are:
- they are easy to use and application does not require a lot of time,
- they can be applied in different stages of the design process, including early stages where the design is not completely defined yet and where cost information is very desirable,
- to some extent the methods are design oriented per definition as they explicitly show main cost influencing parameters and their weights.
The main disadvantage of the methods is that sufficient information on comparable designs has to be available. What is considered 'comparable' depends on the extent of detail of the class definitions, which, in its turn, is related to the potential accuracy of the estimates.

7
Characteristic for detailed cost estimation is that costs for a new object are estimated based on the means that are required to manufacture it. These means include materials, tools, machines and operators. Costs for the latter two are estimated using rates (costs/hour) and required operation times. Essentially, basic cost information is prepared per type of operation.
The main advantages of detailed cost estimation are:
- potential high accuracy,
- data on existing comparable designs are not required,
- it represents a more general and more basic approach than global cost estimation and is closer related to actual production,
- it is already institutionalized in many companies, although not in a design oriented way. Experience and a lot of data are available.
The main disadvantages are:
- a completely detailed description of the design is necessary. The method can only be applied in the last phase of the design process,
- application requires a completely detailed process plan that can not be made by the designer,
- application requires some experience, search for basic data, easy but tedious calculations and quite a lot of time. Consequently the method can only be used by the designer if its application is automated to a considerable extent.

8

Characteristic for design rules and other manufacturability information is that they explicitly or implicitly indicate advantageous design solutions or changes in the design that will lead to lower costs, without specifying costs or savings. The advantages of this type of qualitative cost information are:
- it can avoid the need to estimate costs. Quantitative cost information is replaced by heuristics (the experience that some design solutions are better or worse than others) and by rules that are derived by analysis of manufacturing techniques,
- it is company independent to a considerable extent (although not completely) and it is hardly subject to change in time (it depends primarily on changes in manufacturing technology),
- it is available on different levels of detail. Consequently it can be applied in all stages of the design process,
- it can be supplied passively (by presenting design rules on request) or actively (by testing a design for manufacturability),
- it is design oriented per definition.

Some disadvantages are:
- for some design rules it is not well defined when they may be used and what the reason for their existence is. This incorporates a risk and makes it difficult to keep them up-to-date,
- detailed rules usually depend on the type of operation and sometimes on the detailed way in which the operation is performed. This implies that at least some knowledge of the process plan is required before an object can be tested in detail for manufacturability,
- the information does not quantify costs.

9

Most research on computerized cost information for designers has been focused on the development of tools and techniques for the derivation, maintenance and application of global cost estimation methods. Consequently a fairly complete set of techniques for the development of such methods is available. Some programs for the use of these methods in CAD-environments have been developed. Most of them are limited prototypes (often for turned parts) that illustrate that the approach works. The development and application of such tools in practice does not seem to be widespread however. Basic time data for operations, and techniques to determine them, are available. Detailed cost estimation tools that use these data are available as
well. Most of them are intended for use by process planners or cost estimators however, and they require the corresponding knowledge, time and experience. Usually they are not linked to a CAD-system and the results are not design oriented. Consequently their usefulness for designers is limited. Design rules and other manufacturability information are available on paper, but large data collections in computers for use by designers in CAD-systems seem to lack. Some attempts have been made to control manufacturability during design in CAD, or to test designs automatically for manufacturability. These programs are prototypes that clarify problems and possibilities. Work in this domain is just starting and a lot remains to be done.

10

There are three basic difficulties in providing cost information to the designer:
- most quantitative cost information is time and company dependent,
- the need for cost information is highest in early phases of the design process, where it is most difficult to supply,
- providing cost information costs time and money. These resources are scarce where the information is needed most.

11

Quantitative cost information can be made design oriented by showing the main cost influencing parameters and their weights. This possibility is limited to global cost estimation and the resulting indications for the designer may be rather trivial. Another very attractive possibility is to specify costs per (form) feature. In many cases this is difficult and somewhat artificial however. The difficulties increase with the extent of integration of the manufacturing process. If many features are manufactured simultaneously in an integrated tool, it is difficult to assign costs to the individual features. From this point of view manufacturability information offers advantages. The topic 'how to make cost information design oriented' requires more attention.

12

The development of extensive computerized cost information tools for designers is difficult because some major associated problems have only been solved partly. These are:
- the development of database systems that conveniently handle the storage of the large variety of types of data and of the complex relations between these data that are found in engineering applications. Research on for instance object-oriented databases is ongoing, but additional work is required,
- the development of generative computer aided process planning systems that conceive a design as adaptable. These systems are necessary because detailed cost estimation and (to some extent) generation of manufacturability information require process planning data as input. The designer can not always be expected to supply these data. CAPP is an important international research topic and some advances have been made, but in general progress is still slow.
- the development of CAD-systems based on feature modelling, or the solution of the feature recognition problem. High-level feature descriptions of designs are required for:
  - the presentation of cost information in a design oriented way,
  - the development of automated process planning,
  - automatic testing of a design for manufacturability.
In general the possibilities of features, and the problems associated with their use, should be studied. Sets of features for specific domains have to be developed and attention has to be paid to the difference between design features and process planning features. Also here, work is well underway, but a lot remains to be done,
- the use of Artificial Intelligence techniques for engineering applications. It is probable that the use of AI-techniques will be advantageous when trying to solve problems like process planning, generation of manufacturability information, feature recognition, user interface design and classification of objects. Although some potential advantages and disadvantages of AI-techniques can be indicated, they remain vague. Additional experience is required to clarify them.

13
Most research on cost information tools for designers, automated process planning, feature modelling and feature recognition seems to focus on rotational or prismatic parts that are produced in small quantities using machining operations like turning, milling, drilling, and grinding. To a considerable extent, the associated problems are different for sheet metal design and manufacture and for mass production. These topics deserve more attention.

14
A concept for an integrated design and cost optimization environment, called DIDACOE, has been presented. Due to lack of time, the complexity of the problem and the associated problems mentioned in conclusion 12, only part of this concept has been implemented in computer programs (see figure 4.6). The current implementation focuses on detailed cost estimation for a class of sheet metal parts and includes:
- a program for automatic generation of basic feature descriptions for sheet metal parts, based on 2D representations of such parts in the CAD-system Medusa,
- a program for automatic generation of strip layouts, based on a basic feature description of a sheet metal part,
- a program for presentation of generated strip layouts in the CAD-system Medusa,
- a detailed cost estimation program with a module for time estimation for the production of sheet metal parts from strip in progressive dies.
The programs have been implemented on VAX under VMS in Fortran. A large part of the strip layout generation program was implemented in Nexpert Object, an environment for the development of knowledge-based systems.
A general concept for detailed cost estimation has been developed, including:
- products, assemblies and parts organized in a product structure,
- operations, less worked objects and basic objects organized in production paths,
- a black box model for operations that considers scrap, rejection of output, simultaneous production of several objects and addition of material during the operation,
- a grey box model for operations that distinguishes machines, operators, tools and preparative, secondary and main actions,
- a flexible concept for the definition of operation times based on preparation time, secondary time, main time, time allowance factors, machine use factors and human attention factors.

Main restriction of the concept is that operations must have one type of output only. The concept is expected to be suitable for combination with process planning concepts and with operation specific time and tool cost estimation concepts.

A concept for time estimation for the manufacture of sheet metal parts from strip in progressive dies has been developed. An associated concept for tool cost estimation still lacks.

A set of Basic Features for the description of a class of sheet metal parts has been developed. Currently, only flat parts and parts with straight bends can be described using these features. The set of features can be extended but the approach will never be really suitable for the description of parts with very complex deformations. The description is simple, straightforward and suitable as a basis for automatic part analysis.

A concept for automatic generation of strip layouts has been developed. Using an Advanced Feature description of the outer contour of an unfolded sheet metal part, the concept aims at finding potential advantageous layouts by reasoning. The brute force angle variation technique that is usually applied to find optimal layouts for cut out, is used in default cases only. The current concept is a prototype that does not consider all influence factors. It still has some obvious defects, but in general the approach looks promising.

The work offers possibilities for extension to generation of manufacturability information and to automation of tool layout design. It has provided a lot of insight in the use of features, in the use of AI-techniques, in data storage problems and in process planning problems.

The current DIDACOE-implementation completely automatically generates a detailed cost estimate from a 2D representation in the CAD-system Medusa for sheet metal parts produced from strip in progressive dies. Estimation of die costs is not included however.
This result is nice but not very impressive. It is too limited for any practical use and compared to the ideal DIDACOE-concept a lot of essential facilities, like the generation of manufacturability information, are missing. Considering the complexity of the problems and the fact that a lot had to be developed from scratch, this limited result is understandable.

The general impression is that a lot of basic work has been done and that some extensions should be comparatively easy. These include:

- improvement of the strip layout generation program. Points that need further attention are alternative edge models and the uniformity check. In addition bending considerations should be included,

- extending the strip layout program to include generation of manufacturability information,

- implementation of a tool layout design program and a simple module for tool cost estimation,

- extension to other sheet metal manufacturing techniques. It will not be very difficult to add a module for production from coil and to automate cost estimation in the same way as for strip. A time estimation module for production in conventional dies is also easy to write. It will be hard however to automate the associated process planning. Even if this task is left to the designer, it is not evident that it will be possible to derive all required data from CAD. Maybe the designer will have to flag features in the CAD-representation for this purpose. Extension to deep drawing is not easily possible, as the associated parts are not conveniently described using the current features.

If these extensions are made with the objective to develop a program that can be useful in practice, cooperation with a sheet metal manufacturer will be advantageous.

19

The DIDACOE-project provides a good insight in the needs of designers, in the possibilities to develop cost information tools and in the associated problems. Although many problems were not solved, they have been clarified. Maybe this is a more important result than the current implementation.

Extension of the DIDACOE-concept can certainly be useful, but the main objective of such extensions should be to study more general problems like the ones mentioned in conclusions 11 and 12. It seems particularly interesting to study active supply of manufacturability information.
Appendix 1: General data and datastructures

This appendix provides a survey of the general data and datastructures that are used in cost estimation and process planning, as opposed to the sheet metal specific data in the next appendix. Some data that are for program administration only, such as counters and runtime identifications, have not been included. The survey consists of three parts. Data in the first part describe the Main Objects in the product structure. Data in the second part are associated with operations and production paths. The third part deals with the description of materials and shapes.

Some information on the tables and their relations precedes each part. Additional information on the meaning of the tables and their fields can be found in chapter 5. Paragraph 11.3 addresses relations between tables and between records in general. Some details are explained in notes at the end of the appendix. Fieldtypes are indicated by a letter between brackets preceding the field: C=character string, I=integer number, R=real number.

Part 1: Main Objects and Product Structure

Every Main Object has an entry in the MO-table. This table stores all data that are relevant for any type of Main Object. In addition it contains two subtype fields: an Object subtype and an Estimate subtype.

Four possible continuation tables are available depending on the combination of both subtypes: detailed estimated product, detailed estimated subassembly, detailed estimated makepart and direct estimated buypart. Note that tables for other types of objects and estimation methods can easily be added.

Direct relations between any two Main Objects are defined in a separate MOR-table. The complete product structure can be derived from this information.

Main Object (MO)

(C) Unique Identification
(C) Name for User Reference
(I) Record Number
(I) Object SubType (*1)
(I) Estimate SubType (*2)
(I) SubType Record Number
(R) Total Cost per Piece

(C) UserName of first Creator
(C) Group of Creator (*3)
(C) UserName of last Revisor
(C) Date of first creation (dd-mm-yy)
(C) Date of last revision (dd-mm-yy)
(C) Time of last revision (hh:mm:ss)
(C) Data Protection code (*4)

(I) UpToDate (*5)
(I) Complete (*6)

Detailed Estimated Product (DEPRD)

(R) Total Quantity
(R) Batch Quantity
(R) Material Cost per piece
(R) Operation Cost per piece
(R) Tool Cost per piece
(R) Other Cost per piece

(R) Own Total Cost per piece
(R) Own Material Cost per piece
(R) Own Operation Cost per piece (*7)
(R) Own Tool Cost per piece
(R) Own Other Cost per Quantity
(R) Other Cost Quantity

(I) Operation Record Numbers (array)
Detailed Estimated Subassembly (DESA)

(R) Original Total Quantity
(R) Original Batch Quantity
(R) Active Total Quantity
(R) Active Batch Quantity
(I) Total Quantity Source
(I) Batch Quantity Source

(R) Material Cost per piece
(R) Operation Cost per piece
(R) Tool Cost per piece
(R) Other Cost per piece

(R) Own Total Cost per piece
(R) Own Material Cost per piece
(R) Own Operation Cost per piece (*7)
(R) Own Tool Cost per piece
(R) Own Other Cost per piece
(R) Other Cost Quantity

(I) Operation Record Numbers (array)

BuyPart (BP)

(R) Original Total Quantity
(R) Original Batch Quantity
(R) Active Total Quantity
(R) Active Batch Quantity
(I) Total Quantity Source
(I) Batch Quantity Source

(R) Material Cost per piece
(R) Indirect Material Factor

(C) Estimate Type (*9)
(R) Price per Quantity
(R) Price Quantity
(R) Cost per Unit
(R) Number of Units in one BP

Detailed Estimated MakePart (DEMP)

(R) Original Total Quantity
(R) Original Batch Quantity
(R) Active Total Quantity
(R) Active Batch Quantity
(I) Total Quantity Source
(I) Batch Quantity Source

(R) Material Cost per piece
(R) Operation Cost per piece
(R) Tool Cost per piece
(R) Other Cost per Quantity
(R) Other Cost Quantity

(I) Operation Record Numbers (array)
(I) BasicObject Record Numbers (array)
(I) Material Reference (*8)
(I) Shape Reference (*8)

(R) Waste Revenue per batch
(R) Waste Volume per batch (mm3)
(R) Waste Mass per batch (gram)

Main Object Relations (MOR)

(C) Parent Identification
(C) Child Identification
(I) Direct Number of Appearances

Part 2: Operations and Production Paths

Production paths can be specified for detailed estimated main objects. The tables for these objects have a field 'operation record numbers' that points to one or more records in the general Operation-table (OP). This field defines the operations that belong to the production path for the MO. In addition, detailed estimated makeparts have a field 'basic object record numbers' that points to records in the BasicObject-table (BO).

The operation table (OP) specifies all data that are relevant for any type of operation. Among these data is an operation sequence number, that defines the order of the operations in a path. Currently only one subtype is available: detailed estimated operation. Other subtypes, for instance for global estimation, could be added later.
The DEOP-subtype specifies all data that are relevant for any type of detailed estimated operation. It contains references to a related machine, operator, tool, added material object and less worked object (output of the operation). This reference-field contains either a zero (no relations) or the recordnumber of the related unit in the top-table for its type. One subtype is available: sheet metal production in progressive dies (SMOP1 in appendix 2). Many other subtypes can be added later.

The general Machine- and Tool-tables allow the definition of subtypes. One subtype is currently defined for each: Press for Machine, and Progressive Die for Tool. Tables for these subtypes can be found in appendix 2.

**Operation (OP)**

(C) Name for User Reference  
(I) Estimate SubType (*2)  
(I) SubType Record Number  
(I) Sequence Number in path  
(R) Active Total Quantity  
(R) Active Batch Quantity  
(R) Quantity Factor  
(R) Rejection Factor  
(R) Multiplication Factor  
(R) Integrated Cost per batch  
(I) Complete (*8)

**Detailed Estimated Operation (DEOP)**

(I) Operation SubType (*10)  
(I) SubType Record Number  
(R) Operation Cost per batch  
(R) Material Cost per batch  
(R) Total Tool Cost  
(R) Machine Cost per batch  
(R) Operator Cost per batch  
(R) Machine Time per batch (min)  
(R) Operator Time per batch (min)  
(R) Preparation Time per batch (min)  
(R) Secondary Time per batch (min)  
(R) Main Time per batch (min)  
(R) Time Allowance Factors (array)

(I) Machine Reference  
(I) Operator Reference  
(I) Tool Reference (*8)  
(I) AMO Reference  
(I) LWO Reference

**Operator (MAN)**

(I) Level  
(R) Rate (cost/hour)  
(R) Attention Factors (array)

**Added Material Object (AMO)**

(R) Active Total Quantity  
(R) Active Batch Quantity  
(R) Quantity Factor  
(R) Material Cost per batch  
(R) Indirect Material Factor  
(C) Estimate Type (*9)  
(R) Price per Quantity  
(R) Quantity for Price  
(R) Cost per Unit  
(R) Number of Units in one AMO  
(I) Material Reference (*8)  
(I) Shape Reference (*8)

**Tool (TOL)**

(I) Tool SubType (*12)  
(I) SubType Record Number  
(R) Total Tool Cost  
(R) Basic Cost per piece  
(R) Interest Cost per piece  
(R) Repair & Maint. Cost per piece  
(R) Interest Percentage  
(R) Repair & Maintenance Percentage  
(I) Multiplicity  
(I) Required Number of tools  
(R) Tool Life  
(I) Number of backup tools

**Machine (MACH)**

(C) Identification  
(C) Name for User Reference  
(I) Machine SubType (*11)  
(I) SubType Record Number  
(R) Rate (cost/hour)  
(R) Use Factors (array)

appendix 1-3
Less Worked Object (LWO)  Basic Object (BO)

(R) Active Total Quantity  (I) Basic Object Type (*13)
(R) Active Batch Quantity  (I) Frequency
(R) Quantity Factor

(R) Total Cost per piece  (R) Active Total Quantity
(R) Material Cost per piece  (R) Active Batch Quantity
(R) Operation Cost per piece  (R) Quantity Factor
(R) Tool Cost per piece

(I) Material Reference (*8)
(I) Shape Reference (*8)

(R) Material Reference (*8)
(I) Shape Reference (*8)

(R) Estimate Type (*9)
(R) Price per Quantity
(R) Quantity for Price
(R) Cost per Unit

(R) Material Cost per batch
(R) Indirect Material Factor

(R) Number of Units in one BO

Part 3: Material and Shape

Detailed estimated makeparts, basic objects, less worked objects and added material objects have a Material reference and a Shape reference. Unless they are zero, these fields point to records in the central material- and shape-tables (MAT and SHP). This implies that material and shape can be defined for every object separately. If the material type is the same for makepart, less worked objects and basic object, all objects can reference the same record in the MAT-file. It is also possible however to declare for instance fibres and resin as basic objects and to declare the fibre-reinforced material separately for the less worked objects and the makepart. If necessary, changes in material properties due to the manufacturing process could also be reflected by making LWO's point to different material-records. Similar flexibility is offered for shapes.

Complete hierarchies of material- and shape-types can be imagined. They have not been implemented however. At the moment one material subtype (MATD1) and two shape subtypes (SMP1 and STRIP in appendix 2) have been defined. Other types of material- and shape-descriptions can easily be added. The complete basic feature description for sheet metal parts is accessed through the SMP1-table. In principle, the MATD1-table defines the selection of properties from the common material datafile (not shown here) that is stored as part-specific (see 11.2). The MATD1-file is not identical to the common material data file.

The shape-description for sheet metal LWO's is not completely defined yet. It is possible to use a basic feature description, just like the one for finished makeparts. This would imply however, that a lot of information is stored several times: many features will appear in several LWO's. An alternative that saves a lot of space is, to annotate the basic feature description for the makepart. It should be possible for instance to indicate per feature in which operation it is made. This would define the LWO-shapes indirectly. It is likely however that the LWO's also contain temporary features that do not appear in the final part. This problem requires additional study, just like two associated problems: the description of tool-shapes and the relations between these shapes and LWO-features.

Another remark that should be made is, that material prices now depend on the type of material only. In future a special common data file should be created that indicates how material prices depend on material type, shape type and quantity of material.

appendix 1-4
Material (MAT)

(C) Identification
(C) Name for User Reference
(I) Definition (*14)
(I) Description SubType (*15)
(I) SubType Record Number

Material Description Type 1 (MATD1)

(R) Price per Kilo
(R) Waste Price per Kilo
(R) Specific Mass (kg/dm3)
(R) Tensile Strength (N/mm2)
(R) Bridge Width Factor

Shape (SHP)

(C) Identification
(C) Name for User Reference
(I) Description SubType (*16)
(I) SubType Record Number
(R) Volume (mm3)

Notes

*1 0=unknown, 1=product, 2=subassembly, 3=makepart, 4=buypart.
*2 0=unknown, 1=detailed estimation, 2=direct estimation.
*3 Identification for user and user-group in the VMS operating system. The group is important in relation to access rights.
*4 Same as VMS protection codes: S:rwed, O:rwed, G:rwed, W:rwed.
*5 Indicates whether current MO-data are consistent (up-to-date) with current data on MO-elements (value 1) or not (value 0). A value 0 will trigger recalculation before presentation of data.
*6 Indicates whether all data on the object and its elements are available (value 1) or not (value 0).
*7 Own costs are the costs that are added on product- or subassembly level. They exclude the costs on lower levels, such as part-costs.
*8 A value 0 for a X-reference indicates that no X has been specified (yet).
Another value represents the record number for the X in the file of its type.
*9 String indicating which method is used to estimate costs. Reserved values are 'quant' and 'kilo'. Other units can be defined later or runtime by user.
*10 -1=unknown, 0=general unspecified operation, 1=sheet metal production from strip in progressive dies.
*11 -1=unknown, 0=general unspecified machine, 1=press.
*12 -1=unknown, 0=general unspecified tool, 1=progressive die.
*13 -1=unknown, 0=non-physical, 1=physical.
*14 0=user-defined material, 1=standard material from common materials datafile.
*15 -1=unknown, 0=none, 1=set of properties specified in MATD1.
*16 -1=unknown, 0=none, 1=sheet metal shape description of type 1, 2=strip.

appendix 1-5
Appendix 2: Sheet metal specific data and datastructures

This appendix provides a survey of the sheet metal specific data. The structures contain detailed data on sheet metal parts, as an extension to the more general data described in appendix 1. The survey consists of three parts. The first part contains operation-related data. The second part deals with sheet metal shape description and the third part addresses the advanced feature description.

Information on the object-types and attributes can be found in chapters 6, 7 and 8. Relations between individual structures are discussed in chapter 11. Some additional notes have been added at the end of this appendix.

Fieldtypes are indicated by a letter between brackets: C=character string, I=integer number, R=real number

Part 1: Operation-related data

The SMOP1-table contains continuation-data for detailed estimated operations of subtype 1 (see DEOP-table in appendix 1): production of sheet metal parts from strip in progressive dies. Layout Record Numbers indicate specific records in the LAYOUT-table. The Active Layout indicates the layout that has been used for time- and cost-calculation.

The LAYOUT-table contains general information for a strip layout. Row Record Numbers point to entries in the LAYROW-table. This table specifies information per row in the layout.

The PRESS-table is a continuation table for Machines. The PRGDIE-table is a continuation table for Tools (see appendix 1).

### Detailed Operation of type 1 (SMOP1)

<table>
<thead>
<tr>
<th>SMOP1</th>
<th>LAYOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>(I) Number of Defined Layouts</td>
<td>(I) Number of Rows</td>
</tr>
<tr>
<td>(I) Layout Record Numbers (array)</td>
<td>(I) Row Record Numbers (array)</td>
</tr>
<tr>
<td>(I) Active Layout Number</td>
<td></td>
</tr>
<tr>
<td>(R) Number of Strokes per Strip</td>
<td>(I) Final Operation Type (FOTYP) (*2)</td>
</tr>
<tr>
<td>(I) Correction on Number of Strokes</td>
<td>(R) Pitch (mm)</td>
</tr>
<tr>
<td>(R) Number of Parts per Strip</td>
<td>(R) Object Layout Length (mm)</td>
</tr>
<tr>
<td>(I) Correction on Number of Parts</td>
<td>(R) Overlap Length (mm)</td>
</tr>
<tr>
<td>(I) Number of Strokes Per Minute</td>
<td>(R) Minimal required Strip Width (mm)</td>
</tr>
<tr>
<td>(I) Operation Mode (*1)</td>
<td>(R) Bridge Width Object-Object (mm)</td>
</tr>
<tr>
<td>(R) Switch Factor</td>
<td>(R) Bridge Width Object-Edge (mm)</td>
</tr>
<tr>
<td>(R) Cycle Time (min)</td>
<td>(I) Turn Cutting (0=no, 1=yes)</td>
</tr>
<tr>
<td>(R) Input-Output Time (min)</td>
<td>(R) X-coor. Object System Origin (XSO)</td>
</tr>
<tr>
<td>(R) Movement Time (min)</td>
<td>(R) Y-coor. Object System Origin (YSO)</td>
</tr>
<tr>
<td>(R) Potential Movement Time (min)</td>
<td>(R) Actual OuterContour CutLength (mm)</td>
</tr>
<tr>
<td>(R) Switch Time (min)</td>
<td>(R) Material Use Percentage</td>
</tr>
<tr>
<td>(R) Other Secondary Time (min)</td>
<td>Row in a Layout (LAYROW)</td>
</tr>
<tr>
<td></td>
<td>(R) X-offset for row (DXS, mm)</td>
</tr>
<tr>
<td></td>
<td>(R) Y-offset for row (DYS, mm)</td>
</tr>
<tr>
<td></td>
<td>(R) Orientation Angle (rad)</td>
</tr>
<tr>
<td></td>
<td>(I) Mirrored (-1=yes, 1=no)</td>
</tr>
</tbody>
</table>

appendix 2-1
Machine of type 1 (PRESS)

(R) Capacity (kN)
(I) Minimum Strokes Per Minute
(I) Maximum Strokes Per Minute
(I) Maximum Strip Width (mm)
(I) Maximum Tool Length (mm)

Tool of type 1 (PROGDIE)

(I) Number of Steps
(I) Type Positioning Aid (TYPOS) (*3)
(I) Number of Side Stops
(I) Take Out Percentage
(R) Width Allowance Trim.Stops (mm)

Part 2: Shape descriptions

The STRIP-table and the SMP1-table are continuation tables for the SHP-table (see appendix 1). Except for the STRIP-table, all tables in this part belong to the basic feature description for sheet metal parts. As explained in chapter 11 these data are stored for one part at the same time only. This explains that some references between the tables are missing: it is automatically clear that all data in all tables are part of the same description.
The OCE-, HCE, FCE- and FONF-tables are relation-tables.
Note that some tolerance- and condition pointers have been included in the descriptions, but that they are not being used. The tables to which they point do not yet exist as files in the computer.

Shape description type 2 (STRIP)

(R) Thickness (mm)
(R) Length (mm)
(R) Width (mm)
(R) Area (mm²)
(I) Grain Direction Angle (deg)
(I) Thickness Tolerance Pointer
(I) Width Tolerance Pointer

Shape description type 1 (SMP1)

(R) Thickness (mm)
(R) Effective Blank Area (mm²)
(R) Outer Contour Area (mm²)
(R) Outer Contour Length (mm)
(R) Not-OC Cutline length (mm)
(R) Total Bendline length (mm)
(I) Datum Point Number
(I) Thickness Tolerance Pointer
(I) HeatTreat Condition Pointer
(I) Surface Condition Pointer
(I) Edge Condition Pointer
(I) Flatness Condition Pointer

Point (POINT)

(I) Point Identification Number
(I) Point Type (*4)
(R) X-coordinate (object system)
(R) Y-coordinate (object system)

Straight CutLine Segment (LIN)

(I) LIN Identification Number
(I) Startpoint Number
(I) Endpoint Number
(R) Orientation Angle (rad)
(R) Length (mm)
(I) Edge Condition Pointer

Circular CutLine Segment (RAD)

(I) RAD Identification Number
(I) Startpoint Number
(I) Centrepoint Number
(I) Endpoint Number
(R) Orientation Angle Start (rad)
(R) Orientation Angle End (rad)
(I) Direction (*3)
(R) Radius (mm)
(I) Radius Tolerance Pointer
(R) Segment Length (mm)
(I) Edge Condition Pointer

Full CutCircle (CIR)

(I) CIR Identification Number
(I) Centrepoint Number
(R) Diameter (mm)
(I) Diameter Tolerance Pointer
(R) Circumference Length (mm)
(I) Edge Condition Pointer

appendix 2-2
<table>
<thead>
<tr>
<th>Straight BendLine Segment (LIB)</th>
<th>Curved CutLine Segment (CUR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(l) LIB Identification Number</td>
<td>(l) CUR Identification Number</td>
</tr>
<tr>
<td>(l) Startpoint Number</td>
<td>(l) Startpoint Number</td>
</tr>
<tr>
<td>(l) Endpoint Number</td>
<td>(l) Helppoint Number</td>
</tr>
<tr>
<td>(R) Orientation Angle (rad)</td>
<td>(l) Endpoint Number</td>
</tr>
<tr>
<td>(R) Segment Length (mm)</td>
<td>(R) Orientation Angle Start (rad)</td>
</tr>
<tr>
<td>(R) Bend Angle (rad)</td>
<td>(R) Orientation Angle End (rad)</td>
</tr>
<tr>
<td>(l) Bend Angle Tolerance Pointer</td>
<td>(R) Weight helppoint</td>
</tr>
<tr>
<td>(R) Bend Radius (mm)</td>
<td>(R) Segment Length (mm)</td>
</tr>
<tr>
<td>(l) Bend Radius Tolerance Pointer</td>
<td>(l) Edge Condition Pointer</td>
</tr>
</tbody>
</table>

**Outer Contour (OC)**

(l) Perimeter Type (*6)  
(R) Contour Length (mm)  
(R) Enclosed Area (mm²)  
(l) Side (*7)  
(l) Edge Condition Pointer

**Curved CutLine Segment (CUR)**

(l) CUR Identification Number  
(l) Startpoint Number  
(l) Helppoint Number  
(l) Endpoint Number  
(R) Orientation Angle Start (rad)  
(R) Orientation Angle End (rad)  
(R) Weight helppoint  
(R) Segment Length (mm)  
(l) Edge Condition Pointer

**Outer Contour Elements (OCE)**

(l) Outer Contour Sequence Number  
(l) Element Type (*8)  
(l) Element Number in its type

**Curved CutLine Segment (CUR)**

(l) CUR Identification Number  
(l) Startpoint Number  
(l) Helppoint Number  
(l) Endpoint Number  
(R) Orientation Angle Start (rad)  
(R) Orientation Angle End (rad)  
(R) Weight helppoint  
(R) Segment Length (mm)  
(l) Edge Condition Pointer

**Hole (HOLE)**

(l) Hole Identification Number  
(l) Hole Type (*9)  
(R) Contour Length (mm)  
(R) Enclosed Area (mm²)  
(l) Side (*7)  
(l) Edge Condition Pointer

**Curved CutLine Segment (CUR)**

(l) CUR Identification Number  
(l) Startpoint Number  
(l) Helppoint Number  
(l) Endpoint Number  
(R) Orientation Angle Start (rad)  
(R) Orientation Angle End (rad)  
(R) Weight helppoint  
(R) Segment Length (mm)  
(l) Edge Condition Pointer

**Hole Contour Elements (HCE)**

(l) Hole Identification Number  
(l) Hole Contour Sequence Number  
(l) Element Type (*8)  
(l) Element Number in its type

**Curved CutLine Segment (CUR)**

(l) CUR Identification Number  
(l) Startpoint Number  
(l) Helppoint Number  
(l) Endpoint Number  
(R) Orientation Angle Start (rad)  
(R) Orientation Angle End (rad)  
(R) Weight helppoint  
(R) Segment Length (mm)  
(l) Edge Condition Pointer

**Features on Face (FONF)**

(l) FONF Identification Number  
(l) Face Identification Number  
(l) Feature Type (*10)  
(l) Number of feature in its type

**Curved CutLine Segment (CUR)**

(l) CUR Identification Number  
(l) Startpoint Number  
(l) Helppoint Number  
(l) Endpoint Number  
(R) Orientation Angle Start (rad)  
(R) Orientation Angle End (rad)  
(R) Weight helppoint  
(R) Segment Length (mm)  
(l) Edge Condition Pointer

**Part 3 : Advanced Feature Description**

The tables below describe advanced features as used for the description and derivation of edge-models in the Strip Layout Generation Program. The presented data are only stored runtime in Nexpert's objects and attributes. Therefore the table-descriptions are in fact limited Class-descriptions. In Nexpert, objects from different classes are linked by the definition of parent-child relations. These relations are not shown below.

**Angle Group of LIN (AGL)**

(l) AGL Identification Number  
(R) Orientation Angle (rad)  
(l) Suitability  
(R) Total Length (mm)
Line Group of LIN (LGL)

(I) LGL Identification Number
(I) Startpoint Number
(I) Endpoint Number
(I) OC Sequence Number first LIN
(I) OC Sequence Number last LIN
(R) Nett Length (mm)
(R) Total Length (mm)
(R) Orientation Angle (rad)
(I) Suitability
(R) Y-coordinate rank
(R) Y-coordinate value

Gap (GAP)

(I) GAP Identification Number
(I) Identification Number first LGL
(I) Identification Number second LGL
(R) X-coordinate intersection point
(R) Y-coordinate intersection point
(R) First GAP Length (mm)
(R) Second GAP Length (mm)
(R) GAP Angle (mm)

Circle Group of RAD (CGR)

(I) CGR Identification Number
(I) Startpoint Number
(I) Endpoint Number
(I) OC Sequence Number first RAD
(I) OC Sequence Number last RAD
(R) X-coordinate Centrepoint
(R) Y-coordinate Centrepoint
(R) Orientation Angle Start (rad)
(R) Orientation Angle End (rad)
(R) Radius (mm)
(R) Nett Length (mm)
(R) Total Length (mm)
(I) Suitability

Corner Hole (CHL)

(I) CHL Identification Number
(I) CHL Type (*9)
(I) Startpoint Number
(I) Endpoint Number
(I) OC Sequence Number first segment
(I) OC Sequence Number last segment
(R) X-coordinate intersection point
(R) Y-coordinate intersection point
(R) Contour Length (mm)
(R) First Opening Length (mm)
(R) Second Opening Length (mm)
(R) Opening Angle (rad)
(I) Edge Condition Pointer

Straight Edge (SSE)

(I) Edge Identification Number
(I) Edge Type (+1)
(I) Startpoint Number
(I) Endpoint Number
(I) OC Sequence Number first segment
(I) OC Sequence Number last segment
(I) Identification Number Base LGL
(R) Orientation Angle (rad)
(R) Start Difference Angle (rad)
(R) End Difference Angle (rad)
(R) Combined Length (mm)
(R) Intersection Length (mm)
(R) Total Length (mm)
(R) Nett Length (mm)
(I) Number of CHL's
(I) Number of EHL's
(I) Rotated CHL-match (0=no,1=yes)
(I) Rotated EHL-match (0=no,1=yes)
(I) Rotated Uniformity (*11)
(I) Edge Condition Pointer

Circle Interruption (CINT)

(I) CINT Identification Number
(I) CINT Type (*9)
(I) Startpoint Number
(I) Endpoint Number
(I) OC Sequence Number first segment
(I) OC Sequence Number last segment
(R) X-coordinate CentrePoint
(R) Y-coordinate CentrePoint
(R) Contour Length (mm)
(R) Opening Angle (rad)
(I) Edge Condition Pointer

appendix 2-4
Circle Edge (SCE)

(I) Edge Identification Number
(I) Edge Type (+2)
(R) Diameter (mm)
(R) Combined Length (mm)
(R) Total Length (mm)
(R) Nett Length (mm)
(I) Identification Number Base CGR
(I) Number of CINT's
(R) Largest CINT Opening Angle (rad)
(R) Total CINT Opening Angle (rad)
(I) Edge Condition Pointer

Other Edge (OE)

(I) Edge Identification Number
(I) Edge Type (+3)
(I) Startpoint Number
(I) Endpoint Number
(I) OC Sequence Number first segment
(I) OC Sequence Number last segment
(R) Orientation Angle Start (rad)
(R) Orientation Angle End (rad)
(R) Start Difference Angle (rad)
(R) End Difference Angle (rad)
(R) Combined Length (mm)
(I) Rotated Uniformity (*11)
(I) Edge Condition Pointer

Relation Opposite Edges (ROE)

(I) Identification Number first Edge
(I) Identification Number second Edge
(I) Normal Uniformity (*11)
(I) Mirrored Uniformity (*11)
(I) Both Rotated Uniformity (*12)
(I) Parallel (0=no,1=yes)
(R) Normal Distance (mm)
(I) Normal Match CHL (0=no,1=yes)
(I) Normal Match EHL (0=no,1=yes)
(I) Mirrored Match CHL (0=no,1=yes)
(I) Mirrored Match EHL (0=no,1=yes)

Notes

*1 0=single stroke operation, 1=continuous operation.
*2 -1=unknown, 1=CutOut, 2=NormalCutoff, 3=TwoPartCutOff, 4=Bridge-CutOff, 5=OnePartCutOff.
*3 -1=unknown, 1=fixed stop, 2=fixed stop and side stops, 3=trimming stop at first step, 4=two trimming stops at first and last step.
*4 1=start- or endpoint, 2=centrepoint, 3=CUR-helppoint, 5=datum point, 7=other.
*5 0=anticlockwise, 1=clockwise.
*6 1=rectangle, 2=slant, 3=parallelogram, 4=trapezoid, 5=triangle, 6=circle, 9=other parallel.
*7 Direction of description of the contour. Value -1 indicates that the feature is on the lefthandside when walking along the contour in the direction of description (anticlockwise). Value 1 indicates the righthandside (clockwise)
*8 1=LIN, 2=RAD, 3=CIR, 4=CUR, 5=LIB.
*9 No types have been defined yet.
*10 20=hole, 50=face.
*11 0=not uniform, 1=normal uniform, 2=interrupted uniform, 3=doubtful uniform.
*12 Values obtained by multiplication of the rotated uniformity values for both edges separately (0,1,2,3,4,6,9)

appendix 2-5
List of References

All-1 Allen, R.H., Boarnet, M.G., Culbert, C.J. and Savely, R.T. 'Using hybrid expert system approaches for engineering applications' (Engineering with Computers 2, 1987, p.95-110)
Ame-1 American Machinist 'Software packages enhance accuracy and speed of cost estimating' (American Machinist & Automated Manufac., june 1987, p.61-62)
Apg-1 Apgar, H.E. and Daschbach, J.M. 'Analysis of design through parametric cost estimation techniques' (In:Proc.of ICED'87, Boston, august 1987, p.759-766)
Ast-1 Astrop, A. 'Computer system is quick on the draw' (Machinery and Production Engineering, 15 march 1978, p.42-44)
Bau-1 Baumann, G. 'Ein Kosteninformationsystem für die Gestaltungsphase im Betriebsmittelbau' (Dissertation TU München, 1982)
Blu-1 Blume, P. and Burmester, J. 'Integriertes CAD/CAM-system für Stanzteile' (Zeitschrift für Wirtschaftliche Fertigung, 72 (1977) 5, p.219-224)
Bod-1 Bode, K.-H. 'Konstruktions-Atlas' (Hoppenstedt, 1984)
Boe-1 Boes, P. J. M. 'NC- en CNC-revolverpens, een volwassen plantbewerkings-techniek' (Metaalbewerking 46,7,maart 1980,p.175-185)
Boo-1 Boomsma, H. 'Computerprogramma voor kopsprzbehoreningen' (TU-Delft, Industrial Design Engineering, october 1985)
Bra-2 Branson, J.R. 'Materials conservation and improvement in utilisation by computerised techniques' (Sheet Metal Industries, may 1976, p.280-285)
Bre-1 Bremerberger, M. 'Stanzerei Handbuch für Konstrukteure' (Carl Hanser Verlag, München, 1963)
Bur-1 Burch, J.D. 'Cost estimating with uncertainty' (Ind.Eng.7(3)1975,p.34-37)
Bus-1 Busch, W. 'Relativkostenkataloge als Hilfsmittel zur Kostenfrüherkennung' (In: Konstruktionspraxis im Umbruch, VDI-Berichte 347, VDI-Verlag, Düsseldorf, 1979)
Bus-3 Busch, W. and Heller, W. 'Relativkostenkataloge als Hilfsmittel zur Kostenfrüherkennung' (DIN-Mitteilungen 59 (1980) H.1,p.21-26)

LitRef -1-


Cri-1 Centre national d'innovation et de transfert de technologies 'Conception d'Outil à Suivre Assistée par Computer, COSAC' (product description, CRITT, Chalons sur Marne, 1988)


Din-1 Deutsches Institut für Normung 'DIN 8580 teil 2: Fertigungsverfahren, übersicht' (DIN, Beuth Verlag, Berlin, 1983)


Ehr-1 Ehrspiel, K. 'Reduction of product costs in West Germany' (Proc. of ICED '87, Boston, August 1987, p. 796-806)

Ehr-2 Ehrspiel, K. 'Kostengünstig Konstruieren' (Konstruktionenbücher Band 35, Springer Verlag, 1985)


Ehr-7 Ehrspiel, K. and Baumann, G. 'Kurzkalkulation im Betriebsmittelbau' (VDI-Z 125, Dec. 1983, nr. 23/24, p. 989-993)

Ehr-8 Ehrspiel, K. 'Genauigkeit, Gültigkeitsgrenzen, Aktualisierung der Erkenntnisse und Hilfsmittel zum kostengünstigen Konstruieren' (Konstruktion 32, 1980, H. 12, p. 487-492)


Emm-1 Emmerik, M. J. G. M. and Jansen, F. W. 'Feature Modelling, link tussen CAD en CAM' (CAD/CAM in Bedrijf, June 1988, no. 4, p. 34-37)
Esm-1 Esmeyer, E. in 'Syllabus themdag Artificial Intelligence' (Mikrocentrum West, Delft, september 1987)
Fau-1 Faux, J.D. and Pratt, M.J. 'Computational geometry for design and manufacture' (Ellis Horwood Limited, John Wiley & Sons, Chichester)
Fis-1 Fischer, W. 'Rationelle Erstellung von Angeboten' (Dissertation, RWTH Aachen, 1977)
Fle-1 Flemming, M. 'Qualification of CAD-systems connected to other disciplines' (In:Proc.of ICED'87, Boston, august 1987,p.379-393)
For-1 Foral, E. 'Rechnerunterstützte Blechtailekonstruktion' (VDI-Berichte 523, Konstruirer mit Blech, Düsseldorf, VDI-Verlag 1984, p. 125-140)
Gei-1 Geiser, W. 'Fertigungstechnik 2' (Verlag Handwerk und Technik, 1978)
Haa-1 Haan, T.J.A. 'Cost estimating during the design process' (Proc.of ICED'83, volume 1, Kopenhagen 1983, p.174-)
Haa-2 Haan, T.J.A. 'Costprice calculation for parts of sheet metal' (Proc.of ICED'85, volume 1, Hamburg 1985, p.495-)
Han-1 Hansen, F. 'Konstruktionssystematik' (VEB-Verlag Technik, Berlin, 1968)
Har-1 Hartingsveidt, C. van and Voamer, J. 'CAD/CAM koppeling II, multi client study: consequenties voor de konstrukteur ten aanzien van verspanend te bewerken produkten'(Metaal Instituut TNO, 88M/07505/HAR, june 1988)
Her-1 Hermans, H. 'Costprice calculation for mass production' (Proc.of ICED'85, volume 1, Hamburg 1985, p.488-)

LitRef -3-
Her-2 Hermans,H. 'Vervaardigingstechnieken 3, deel I: ponsen en buigen' (Course material for 1027, TU-Delft, Industrial Design Engineering, 1979)
Her-3 Hermans,H. 'Voorkalkulatie van gereedschappen' (course material for 1069, 'Technologie van de maasproductie', august 1982, TU-Delft, Industrial Design Engineering)
Her-4 Herik,J. van der 'Inleiding Artificial Intelligence technieken' (Course material for subject a222, feb.1987, TU-Delft, Information Sciences)
Hil-1 Hillebrand,A. and Ehrienspiel,K. 'Suchkalkulation, ein Hilfsmittl zum kostengünstigen Konstruieren' (Proc. ICED'85 vol.1,Hamburg, 1985,p.451-)
Iwa-1 Iwata,K. and Fukuda,Y. 'KAPPS: know-how and knowledge assisted production planning system in the machining shop' (Computer Aided Process Planning, 19th CIRP Int.Sem.on Manufacturing Systems, Penn State, USA, june 1987, p.287-294)
Kem-1 Kemna,R. 'Energiebewust ontwerpen' (TU-Delft, Industrial Design Engineering, io bol, 1987)
Kes-1 Kesselring,F. 'Sinnvolle Technik' (Konstruktion,H.5,1954,p.169-)
Kie-1 Kiewert,A. 'Systematische Erarbeitung von Hilfsmitteln zum kostenarmen Konstruieren' (TU München, dissertation, 1979)
Kni-1 Knight,W.A. and Poli,C. 'A systematic approach to forging design' (Machine Design 2, jan. 1985, p.94-99)
Kni-2 Knight,W.A., Motherway,C.E. and Poli,C. 'Computer-aided design for increased productivity' (Proc. of ICED'83, Copenhagen 1983, p.639-)
Koo-1 Koolschijn,J. and Wijnberg,H.E. 'Kostencalculatie- en Waarderingsproblemen' (Samson,Alphen a/d Rijn, 1977)
Kre-1 Kreisfeld,P. 'Kostenbestimmung mit CAD-systemen für Rotationsteile' (Carl Hanser Verlag, München, Wien, 1985)
Lim-1 Lim,B.S. 'An IKBS for integrating component design to tool engineering' (Expert Systems, november 1987, vol.4, no.4,p.252-281)
Mah-1 Mahmoud,M.A.M. and Pugh,S. 'The costing of turned components at the design stage' (in:'Information for designers',4th Int.Symp. Univ. of Southampton, july 1979, p.37-42)

LitRef -4-
Pahl, G. and Beelrich, K.H. 'Konstruktionslehre' (Springer Verlag, Berlin, 1976)
Paraszcz, D.L. 'Why engineers should not use Artificial Intelligence' (Infor, vol.26, no.4, 1988, p.234-246)
Picke1, H. and Ehrlienspiel, K. 'Berücksichtigung der Modellkosten bei der Konstruktion von Gussteilen' (Proc. of ICED'85, Hamburg 1985, p.468-)
Picke1, H. 'Wendeschneidwerkzeuge' (Werkstatt und Betrieb 103 (1970) 6, p.421-427)
Priko1, P. 'Die and stamping estimating' (Machine and Tool Blue Book, eight short articles published one each month from march to nov. 1976)
Radmeracher, W. 'Entwicklung eines KostenInfosystems für die Konstruktionsbereich' (Dissertation, TH Aachen, 1980)
Rank, G. and Vergeest, J. 'Utwisseling van product model data: verwachingen op korte en lange termijn' (Ontwikkelingen rond technische automatisering, Congreewerk, Kdrijden CAPE'87, Samsom, Alphen a/d Rijn, 1987)
Rau, T. 'Kostenoptimierung konstruktiver Lösungen: Möglichkeiten für die Einzel- und Kleinserienproduktion' (VDI Taschenbücher T31, VDI-Verlag, Düsseldorf, 1978)
Rehak, R. and Howard, H.C. 'Interfacing expert systems with design databases in integrated CAD systems' (Computer Aided Design, vol.17, no.9, nov. 1985, p.443-449)
Richer, W. 'Kostenbetrachtungen in Konstruktionsmethoden' (Konstruktion 30 (1978) H.12, p.493-496)
Rieg, F.P. 'Kostenwachstumsgesetze für Baureihen - ein Beitrag zur Konstruktionslehre' (Dissertation, Darmstadt, 1982)
Rutz, A. 'Konstruieren als gedanklicher Proess' (Dissertation, TU-München, 1985)
Schulze, G.E. 'Relativkostenfaktoren in der metallverarbeitenden Industrie' (Fortgeschrittliche Betriebsführung/Ind.Eng. 30 (1981)2, p.86-
Schuppar, H. 'Rechnerunterstützte erstellung und aktualisierung von Relativkosten-kataloge' (Dissertation RWTH Aachen, 1977)
Schouwenberg, H.J.E. 'Toepassing van programmeerbare rekenmachines bij construieren van spuitgietmatrizen' (Metaal & Kunststof 21 (1983)5, p.18-25)
Schouwenberg, H.J.E. 'De kostprijs van een kunststof spuitgietproduct' (Metaal & Kunststof 21 (1983) nr.24, p.36-39)
Sch-5 Schouenber, H.J.E. 'Kostenverlaging door etagematrijzen' (Metaal & Kunststof 22 (1984) nr.12, p.16-17)
Sch-7 Schouenber, H.J.E. 'Economisch produceren van kunststof spuitgietproduken' (Metaal & Kunststof 25 (1987) nr.24/25, p.20-23)
Sch-8 Schouenber, H.J.E. 'Is standaardisering van spuitgietmatrijksystemen mogelijk?' (Metaal & Kunststof 4, 22 feb.1988, p.52-55)
Sch-10 Schlosser, D. 'Hydraulische Pressen für die Blecheinbehandlung' (VDI-Z., Bd.126 (1984), nr.15-16 (august), p. M107-M137)
Sch-11 Schrauwers, A. 'Werkvoorbereiding laat zich lastig automatiseren' (Ingenieur en Computer (appendix to 1-kwadraat), 8, aug.1989, p.20-25)
See-1 Seemuller, K.-H. 'Zeitermittlung für Ausnahme- arbeiten an Pressen' (Werkstattblatt 551, Carl Hanser Verlag, München, 1972)
Ste-1 Steeman, W. 'Milieu-aspekten van het ontwerpen' (TU-Delft, Industrial Design Engineering, graduation report, nov.1986)
Sto-1 Stomph-Blessing, L.T.M. 'Methodisch ontwerpen en kunstmatige intelligentie, een literatuur-onderzoek' (TU-Twente, Faculteit der Werktuigbouwkunde, Leerstoel Ontwerp- en Constructieleer, sept.1988)
Sub-1 Subramanyam, S., Lu, S.C-Y. and Zdeblick, W.J. 'A characterization of the process planning task from an artificial intelligence perspective' (Computer Aided Process Planning, 19th CIRP International Seminar on Manufacturing Systems, Penn State, USA, june 1987, p.197-208)
Tas-1 Tasnari, R. 'La matrise des couts industriels' (Les éditions d'organisation, Paris, 1981)
Tim-1 Timmer, H. 'Flexibele productie van onderdelen van dunne metaalplaat' (TU-Delft, Mechanical Engineering, Vervoeringskunde, student report no.88.014, march 1988)
Ull-1 Ullmann, D.G. 'Mechanical Design Methodology: implications on future developments of Computer Aided Design and Knowledge-based Systems' (Engineering with Computers 2, 1987, p.21-29)
Vdi-1 VDI: Verein Deutscher Ingenieure 'Wirtschaftliche Grundlagen für den Konstrukteur' (VDI Richtlinie 2234 Entwurf, VDI-Verlag, Düsseldorf, 1982)
Vdi-3 VDI: Verein Deutscher Ingenieure 'Erstellung und Anwendung von Konstruktionskatalogen' (VDI-Richtlinie 2222, VDI-Verlag, Düsseldorf, 1982)
Vdi-4 VDI: Verein Deutscher Ingenieure 'Technisch-Wirtschaftliches Konstruieren' (VDI-Richtlinie 2225, VDI-Verlag, Düsseldorf, 1977)
Vdi-5 VDI: Verein Deutscher Ingenieure 'Kostenrechnung mit Maschinenstundensätzen' (VDI-Richtlinie 3258, VDI-Verlag, Düsseldorf, 1962)
Ver-1 Vernon, I. R. 'Realistic cost estimating for manufacturing' (Soc. of Manufacturing Engineers, Dearborn, Michigan, 1968)
Vro-1 Vroom, R. 'Toepasbaarheid databases in techniek nog niet probleemloos' (Ingenieur en Computer, no.5, september 1988, p.36-39)
War-1 Warnecke, H. J., Bullinger, H. J. and Hichert, R. 'Kostenrechnung für Ingenieure' (Carl Hanser Verlag, 1981)
War-3 Warnecke, G. and Mertens, P. 'Auswahlkriterien für Expertensystem-Entwicklungsarbeiten' (Zeitschr. für wirtschaftl. Fertigung 82(1987),p.82-86)
Wee-1 Weege, R. D. 'Recyclinggerecht Konstruieren' (VDI, Düsseldorf, 1981)
Wie-4 Wierda, L. S. 'Product cost estimation by the designer' (Engineering Costs and Production Economics 13, 1988, p.189-198)
Wie-7 Wierda, L. S. 'Interpretatie gekoppeld aan CAD-systeem' (CAD/CAM in Bedrijf, no.7, okt. 1988, p.82-85)
Wie-8 Wierda, L. S. 'Van CAD naar CAPP, het interpretatieprobleem' (CAD/CAM in Bedrijf, no.4, juni 1988, p.63-69)
Wie-9 Wierda, L. S. 'A CAD Interpretation Program for Recognition of Features in 2D representations of Sheet Metal Parts' (TU-Delft, Industrial Design Engineering, internal memo K185, february 1989)
Wie12 Wierda, L. S. 'Keuze van een ontwikkelomgeving voor kennisystemen' (PT Industriële Automatisering, no.5, mei 1989, p.60-64)
Wil-1 Wilson, F. W. 'Manufacturing planning and estimating Handbook' (Society of Manufacturing Engineers, McGraw Hill, 1963)
Win-1 Winston, P. H. 'Artificial Intelligence' (Addison Wesley, 1984)
Wit-1 Witte, J. J. S. C. de and Kwee, A. Y. L. 'Kennisystemen in de industrie' (Samsom, Alphen a/d Rijn, 1988)


Ydo-1 YDO normenservice BV 'YDO Calc' (Productinformation 1986, YDO, Postbus 165, 2742 AD Waddinxveen)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACBQ</td>
<td>Active Batch Quantity (5.4)</td>
</tr>
<tr>
<td>ACTQ</td>
<td>Active Total Quantity (5.4)</td>
</tr>
<tr>
<td>AGL</td>
<td>Angle Group of LIN (8.7.2)</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence (9.1)</td>
</tr>
<tr>
<td>AMO</td>
<td>Added Material Object (5.3)</td>
</tr>
<tr>
<td>ASCII</td>
<td>American Standard Code for Information Interchange</td>
</tr>
<tr>
<td>BIOT</td>
<td>Basic Input Output Time In minutes/action (6.5.2)</td>
</tr>
<tr>
<td>BO</td>
<td>Basic Object (5.2)</td>
</tr>
<tr>
<td>BOF</td>
<td>Basic Object Frequency (5.2)</td>
</tr>
<tr>
<td>BP</td>
<td>BuyPart (5.1)</td>
</tr>
<tr>
<td>BQ</td>
<td>Batch Quantity (5.4)</td>
</tr>
<tr>
<td>BQS</td>
<td>Batch Quantity Source (5.4)</td>
</tr>
<tr>
<td>BREP</td>
<td>Boundary Representation</td>
</tr>
<tr>
<td>BWOO</td>
<td>Bridge Width in mm (6.3)</td>
</tr>
<tr>
<td>BWOOF</td>
<td>Bridge Width Factor (6.3)</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer Aided Design</td>
</tr>
<tr>
<td>CAM</td>
<td>Computer Aided Manufacturing</td>
</tr>
<tr>
<td>CAPP</td>
<td>Computer Aided Process Planning</td>
</tr>
<tr>
<td>CDF</td>
<td>Cost Data File (11.1)</td>
</tr>
<tr>
<td>CE</td>
<td>Circle Edge (8.7.2)</td>
</tr>
<tr>
<td>CCR</td>
<td>Circle Group of RAD (8.7.2)</td>
</tr>
<tr>
<td>CHL</td>
<td>Corner Hole (8.7.4)</td>
</tr>
<tr>
<td>CINT</td>
<td>Circle Interruption (8.7.2)</td>
</tr>
<tr>
<td>CIR</td>
<td>Full Cut Circle (7.2.2)</td>
</tr>
<tr>
<td>COR</td>
<td>Corrected-file (7.3.1, 11.1)</td>
</tr>
<tr>
<td>CSG</td>
<td>Constructive Solid Geometry</td>
</tr>
<tr>
<td>CT</td>
<td>Cycle Time in minutes (6.5.1)</td>
</tr>
<tr>
<td>CUR</td>
<td>Curved Cutline segment (7.2.2)</td>
</tr>
<tr>
<td>DARS</td>
<td>Data Access Routines (7.3.3)</td>
</tr>
<tr>
<td>DAT</td>
<td>Common Data-file (11.1)</td>
</tr>
<tr>
<td>DNA</td>
<td>Direct Number Of Appearances (5.1)</td>
</tr>
<tr>
<td>DXS</td>
<td>X-distance between rows (8.3.2)</td>
</tr>
<tr>
<td>DYS</td>
<td>Y-distance between rows (8.3.2)</td>
</tr>
<tr>
<td>EHL</td>
<td>Edge Hole (8.7.2)</td>
</tr>
<tr>
<td>FACE</td>
<td>Bend Face (7.2.1, 7.2.3)</td>
</tr>
<tr>
<td>FOTYP</td>
<td>Final Operation Type (6.2.5)</td>
</tr>
<tr>
<td>GAP</td>
<td>Gap between LGL's (8.7.4)</td>
</tr>
<tr>
<td>HAF</td>
<td>Human Attention Factor (5.7.1)</td>
</tr>
<tr>
<td>IGES</td>
<td>Initial Graphics Exchange Specific.</td>
</tr>
<tr>
<td>IOT</td>
<td>Input Output Time min/part (6.5.2)</td>
</tr>
<tr>
<td>KBS</td>
<td>Knowledge-Based System (9.1.3)</td>
</tr>
<tr>
<td>LGL</td>
<td>Line Group of LIN (8.7.2)</td>
</tr>
<tr>
<td>LIB</td>
<td>Straight line segment (7.2.2)</td>
</tr>
<tr>
<td>LIN</td>
<td>Straight cutline segment (7.2.2)</td>
</tr>
<tr>
<td>LWO</td>
<td>Less Worked Object (5.2)</td>
</tr>
<tr>
<td>M</td>
<td>Tool Multiplicity (6.2.5)</td>
</tr>
<tr>
<td>MIR</td>
<td>Mirrored (8.3.2)</td>
</tr>
<tr>
<td>MO</td>
<td>Main Object (5.1)</td>
</tr>
<tr>
<td>MP</td>
<td>MakePart (5.1)</td>
</tr>
<tr>
<td>MT</td>
<td>Main Time in min/part (5.7.1)</td>
</tr>
<tr>
<td>MVT</td>
<td>Movement Time in min/part (6.5.2)</td>
</tr>
<tr>
<td>MUF</td>
<td>Machine Use Factor (5.7.1)</td>
</tr>
<tr>
<td>NEXPERT</td>
<td>Software of Neuron Data (9.4)</td>
</tr>
<tr>
<td>NOR</td>
<td>Number of Rows in a layout (6.3)</td>
</tr>
<tr>
<td>NSS</td>
<td>Number of Strokes per Strip (6.4)</td>
</tr>
<tr>
<td>NSTEPS</td>
<td>Number of steps in a die (6.1)</td>
</tr>
<tr>
<td>NXP</td>
<td>Nexpert file format (7.3.2, 11)</td>
</tr>
<tr>
<td>OBQ</td>
<td>Operation Batch Quantity (5.4)</td>
</tr>
<tr>
<td>OC</td>
<td>Outer Contour (7.2.1)</td>
</tr>
<tr>
<td>OE</td>
<td>Other Edge (8.7.4)</td>
</tr>
<tr>
<td>OLL</td>
<td>Object Layout Length in mm (6.3)</td>
</tr>
<tr>
<td>O LW</td>
<td>Object Layout Width in mm (6.3)</td>
</tr>
<tr>
<td>OMF</td>
<td>Multiplication Factor (5.3, 6.4)</td>
</tr>
<tr>
<td>OP</td>
<td>Operation</td>
</tr>
<tr>
<td>ORANG</td>
<td>Orientation angle in rad. (8.3.2)</td>
</tr>
<tr>
<td>ORBQ</td>
<td>Original Batch Quantity (5.4)</td>
</tr>
<tr>
<td>ORTQ</td>
<td>Original Total Quantity (5.4)</td>
</tr>
<tr>
<td>OTQ</td>
<td>Operation Total Quantity (5.4)</td>
</tr>
<tr>
<td>PITCH</td>
<td>Strip movement distance in mm (6.3, 8.3.2)</td>
</tr>
<tr>
<td>PMTYP</td>
<td>Perimeter Type (8.9.2)</td>
</tr>
<tr>
<td>PMVT</td>
<td>Potential Movement Time in minutes/movement</td>
</tr>
<tr>
<td>PRD</td>
<td>Product (5.1)</td>
</tr>
<tr>
<td>PT</td>
<td>Preparation Time min/batch (5.7.1)</td>
</tr>
<tr>
<td>RAD</td>
<td>Circumference segment (7.2.2)</td>
</tr>
<tr>
<td>RJS</td>
<td>Rejection Factor (5.3)</td>
</tr>
<tr>
<td>Q</td>
<td>Quantity</td>
</tr>
<tr>
<td>SA</td>
<td>SubAssembly (5.1)</td>
</tr>
<tr>
<td>SDF</td>
<td>Strip Dimension Factor (6.5.2)</td>
</tr>
<tr>
<td>SF</td>
<td>Switch Factor (6.5.2)</td>
</tr>
<tr>
<td>SL</td>
<td>Strip Length in mm (6.3)</td>
</tr>
<tr>
<td>SMD</td>
<td>Sheet Metal Design module (7.3.1)</td>
</tr>
<tr>
<td>SPM</td>
<td>Strokes Per Minute (6.5.4)</td>
</tr>
<tr>
<td>SSE</td>
<td>Suitable Straight Edge (8.7.2)</td>
</tr>
<tr>
<td>ST</td>
<td>Secondary Time in min/part (5.7.1)</td>
</tr>
<tr>
<td>SW</td>
<td>Strip Width in mm (6.3)</td>
</tr>
<tr>
<td>T</td>
<td>Sheet Thickness in mm</td>
</tr>
<tr>
<td>TAF</td>
<td>Time Allowance Factor (5.7.1)</td>
</tr>
<tr>
<td>TENS</td>
<td>Tensile Strength in N/mm2 (6.5.4)</td>
</tr>
<tr>
<td>TQ</td>
<td>Total Quantity (5.4)</td>
</tr>
<tr>
<td>TQS</td>
<td>Total Quantity Source (5.4)</td>
</tr>
<tr>
<td>TYP</td>
<td>Type of Positioning aid (6.2.5)</td>
</tr>
<tr>
<td>VAX</td>
<td>Type of computer of Digital Equipment Corporation</td>
</tr>
<tr>
<td>VMS</td>
<td>Type of operating system of Digital Equipment Corporation</td>
</tr>
<tr>
<td>XSO</td>
<td>X-position object system origin in strip coordinate system (8.3.2)</td>
</tr>
<tr>
<td>YSO</td>
<td>Y-position object system origin in strip coordinate system (8.3.2)</td>
</tr>
</tbody>
</table>
Curriculum Vitae