Computer Supported Design of Production Machines
Computer Supported Design of Production Machines

Barry H. de Roode

31 oktober 1999
1

Het conceptuele ontwerpproces wordt volledig beschreven door het productmodel, ontwerpectiviteiten en de database. Formele modellen van deze elementen vormen een noodzakelijke basis voor computerondersteuning in deze fase van het ontwerpproces.

2

Als jongeren onder 18 jaar geen rijbewijs mogen bezitten, dan zou het ook ouderen boven 75 jaar verboden moeten worden.

3

Het aantal calorieën dat een recreatieve voetballer tijdens een wedstrijd verliest, is minder dan het aantal calorieën dat hij doorgaans in de 90 minuten na de wedstrijd consumeert.

4

Algemene productmodellen, vaak binnen de academische wereld ontwikkeld, kunnen specifieke producten slechts algemeen beschrijven en missen daardoor een vruchtbare praktische utilisatie. Deze productmodellen kunnen echter wel als basis dienen om specifieke productmodellen te ontwikkelen.

5

Historisch gezien zijn geniale personen vaak wetenschappers met lang haar. Er bestaat echter geen rechtstreeks verband tussen genialiteit en het beroep van wetenschapper.
6

Onderzoek naar ondersteuning van het ontwerpproces door middel van computers zal alleen dan vooruitgang kunnen boeken als tijdens het onderzoek een poging tot implementatie gemaakt wordt.

7

Het lezen van bestaande stellingen met het doel om tot een nieuwe en originele stelling te komen, vermindert het aantal potentiële stellingen.

8

Pink Floyd's gitarist David Gilmour laat horen dat het mogelijk is om met een pentatonische toonladder en de juiste timing een virtuoze gitaarsolo te creëren.

9

Voordat informatie over vorige ontwerpen tijdens het ontwerpproces hergebruikt kan worden, zal eerst bekend moeten zijn hoe deze informatie beschreven kan worden en hoe deze tijdens het ontwerpproces gegenereerd wordt.

10

Deze stelling heeft, als zesde stelling die geen betrekking heeft op het onderwerp van dit proefschrift, een cruciale functie in de totstandkoming van dit proefschrift.
Computer Supported Design of Production Machines

Barry de Roode
Computer Supported Design of Production Machines

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K. P. Wakker,
in het openbaar te verdedigen ten overstaan van een commissie,
door het College voor Promoties aangewezen,

op dinsdag 30 november 1999 te 13.30 uur

door

Barry Hendrik DE ROODE

werktuigkundig ingenieur,
geboren te Vlaardingen.
Dit proefschrift is goedgekeurd door de promotoren:
Prof. ir. H.A. Crone
Prof. dr. ir. K. van der Werff

Samenstelling promotiecommissie:

Rector Magnificus
Prof. ir. H.A. Crone
Prof. dr. ir. K. van der Werff
Prof. Dr.-Ing. habil. W. Gerhardt
Prof. dr. I. Horváth
Prof. ir. P. de Ruwe

voorzitter
Technische Universiteit Delft, promotor
Technische Universiteit Delft, promotor
Technische Universiteit Delft
Technische Universiteit Delft
Technische Universiteit Delft

Copyright © 1999 by B.H. de Roode
All rights reserved

ISBN 90-9013297-X
Acknowledgements

Writing a dissertation is not an easy thing to do. Especially the months before the deadline are very stressful and it looks as if no one can help you. When you have finally finished, you have the feeling that you did it all by yourself. This feeling remains until you start writing the acknowledgements. You begin to realise that many people supported you the last four years. Some gave comments on your work, some cooked you a nice meal, and others just asked how things were going. They all directly and indirectly contributed to the book that you are currently reading. Therefore, I have to be grateful to many people and to some in particular.

My supervisors, Prof. ir. H.A. Crone, Prof. dr. ir. K. van der Werff, and Prof. Dr.-Ing. habil. W. Gerhardt are gratefully acknowledged for their support during the last four years. The students that participated in the project are thanked for their work and pleasant co-operation.

I would like to thank my colleagues at the Laboratory for Production Engineering and Industrial Organisation for having humorous conversations, playing golf, riding carts, and for winning the first price at the Delft University of Technology Lustrum Pentathlon 1997. I especially want to thank my daily colleagues André Hoogstrate, Marcel Tichem, Hans-Willem van Vliet, and Jeroen Vos.

Special words of thank go to my parents, Wanda, Jeroen, Gita, Thomas, and all my friends. These people inspired me, often without them knowing, to carry on during the week, weekends, and nights.

Barry de Roode

Vlaardingen
October, 1999
Summary

The conceptual design stage is hardly supported by tools, especially when the design of production machines is regarded. Consequently, designers are often thrown on their own resources while they create designs. Therefore, tools must be developed that support the designer during the conceptual design stage. The goal of this research is to investigate how designers of production machines can be supported during the conceptual stage of the design process.

The research approach was as follows. A literature study has been performed to identify the deficiencies and opportunities of current product models, models of the design process, and computer tools that support design. Based on these results, a generic model of the design process has been developed. This model has been applied to the design of production machines and has been implemented into a design support system. Several case studies have been performed, resulting in conclusions and recommendations for future research.

The developed model of the design process consists of three elements: a product model describes the product to be designed, design activities describe the operations that create this product model, and the database contains background information that can be consulted during design. Each of these elements must be supported by a design support system and therefore, a meta product model, procedures of design activities, and a data schema are developed. The meta product model supports the documentation of a product model that consists of multiple aspect models, each highlighting the design from a particular point of view. The procedures of design activities support the execution of actual design activities by defining how activities are executed, which input they require, and which output they generate. The data schema defines a data structure is used to create the database that stores background information. The generic meta product model, procedures of design activities, and the data schema have been applied to the design of production machines.

The meta product model for production machines consists of five complementary models that describe specific characteristics of production machines. A product state model presents the states that the product to be produced has during the production process. A function model presents the functions that the design must realise and the sequence in which they must be realised. A solution model shows the sub-solutions within the machine and the sequence in which they transform the product. A solution layout model visualises the form and the spatial placement of solutions. Finally, a state
transition model shows the transitions that the solutions perform to realise a function, and how these transitions must be related to each other for a proper functioning of the machine.

The procedures of design activities for production machines are divided in five categories. Administration activities document and presentation activities visualise the product model. The synthesis activities support the designer in finding information in the database and in optimising the product model. The analysis activities check the correctness of relations in the state transition diagram and simulate the working of the machine. The evaluation activities are in current design support system not supported.

The data schema for production machines is used to create a database that contains information about past designs that can be reused in new design cases. Several search queries have been developed that facilitate the easy retrieval of information from the database.

A prototype design support system integrates and implements the developed models and shows their feasibility. It was successfully used to perform a number of case studies in which production machines were modelled by using the system. The results of these cases are promising and indicate that the design support system can assist during the documentation and visualisation of the product model, execution of design activities, and storing and retrieving information from the database. The cases revealed a number of possible improvements to both the prototype and models.

The conclusions of the research are the following. The design process can be divided into a product model, design activities, and the database. Generic models of these elements can be developed and can be applied to a specific application domain, which in this research was the design of production machines. The design support system demonstrated that the specific models can be integrated and implemented. The design support system assists the designer in designing a machine easier, faster, and with fewer errors.

The recommendations of the research are the following. The prototype should be developed further to obtain a prototype that can be evaluated in practice. Several views in the meta product model can be improved and supplementary views can be added. New design activities should be defined and added to the available design activities. The database must be filled with a reasonable amount of information to provide the designer with more alternatives. Further, it must be investigated how the prototype can connect to commercial tools and how the design phases before and after conceptual design can be integrated.
Samenvatting

De conceptuele ontwerpfase wordt nauwelijks ondersteund door hulpmiddelen, vooral wanneer het ontwerpen van productiemachines betreft. Het gevolg is dat ontwerpers tijdens het ontwerpen vaak op hun eigen vindingrijkheid zijn aangewezen. Daarom moeten hulpmiddelen ontwikkeld worden die de ontwerpers tijdens het conceptueel ontwerpen ondersteunen. Het doel van dit onderzoek is om te bepalen hoe ontwerpers tijdens het conceptuele ontwerpproces ondersteund kunnen worden.

De aanpak van het onderzoek was de volgende. Een literatuurstudie is verricht om de tekortkomingen en mogelijkheden te bepalen van de huidige productmodellen, modellen van het ontwerproces, en computerprogramma's die het ontwerproces ondersteunen. Gebaseerd op deze resultaten is een algemeen model van het ontwerproces ontwikkeld. Dit model is toegepast op het ontwerpen van productiemachines en is geïntegreerd in een prototype ontwerpondersteuning. Diverse case-studies zijn uitgevoerd en resulteerden in conclusies en aanbevelingen voor verder onderzoek.

Het ontwikkelde model van het ontwerproces bestaat uit drie elementen: een productmodel dat het te ontwerpen product beschrijft, ontwerpactiviteiten die de operaties die het productmodel creëren beschrijven, en een database die de achtergrondinformatie die tijdens het ontwerpen geraadpleegd kan worden beschrijft. Elk van deze elementen moet ondersteund worden door de ontwerpondersteuning. Om dit te realiseren zijn een meta productmodel, procedures van ontwerpactiviteiten, en een data schema ontwikkeld. Het meta productmodel ondersteund de creatie van een productmodel dat uit verschillende aspectmodellen bestaat. Elk van deze aspectmodellen toont een specifieke beschrijving van het ontwerp. De procedures van ontwerpactiviteiten definiëren hoe activiteiten worden uitgevoerd, wat de invoer moet zijn, en wat de resultaten zullen zijn. Het data schema definiëert een datastructuur die gebruikt kan worden bij de creatie van een database met achtergrondinformatie. Deze algemene modellen zijn toegepast op het ontwerpen van productiemachines.

Het meta productmodel van productiemachines bestaat uit vijf aspectmodellen die specifieke eigenschappen van productiemachines beschrijven. Een product-toestand model toont de toestanden die het te produceren product ondergaat. Een functiemodel beschrijft de functies die het ontwerp moet realiseren en de volgorde waarin ze uitgevoerd moeten worden.
Een oplossingenmodel geeft de oplossingen waaruit het ontwerp bestaat aan, alsmede de volgorde waarin de oplossingen het te produceren product zullen bewerken. Een oplossingen-layout model toont de vorm en de ruimtelijke plaatsing van oplossingen. Als laatste beschrijft een toestand-transitie model de transities die oplossingen ondergaan om een functie te realiseren en hoe deze transities van elkaar moeten afhangen voor een goede werking van de machine.


Het data schema voor productiemachines wordt gebruikt om een database met informatie over vroegere ontwerpen te creëren. Diverse zoekmogelijkheden zijn ontwikkeld om informatie op te vragen.

De prototype ontwerpondersteuning integreert en implementeert de ontwikkelde modellen en toont hun toepasbaarheid. Het prototype werd succesvol gebruikt bij het verrichten van case-studies waarin productiemachines gemodelleerd werden. De resultaten hiervan zijn veelbelovend en duiden erop dat de ontwerpondersteuning daadwerkelijk ondersteund tijdens de documentatie en visualisatie van het productmodel, het uitvoeren van ontwerpactiviteiten, en het opslaan en opvragen van informatie in de database. Daarnaast brachten de case-studies een aantal mogelijk verbeteringen aan het prototype en de modellen aan het licht.

De conclusies van het onderzoek zijn de volgende. Het ontwerpproces kan opgedeeld worden door een productmodel, ontwerpactiviteiten, en een database. Algemene modellen van deze elementen kunnen ontwikkeld worden om vervolgens toegepast te worden op een specifiek applicatiegebied, dat in het geval van dit onderzoek het ontwerpen van productiemachines is. De prototype ontwerpondersteuning toont aan dat de specifieke modellen geïntegreerd en geïmplementeerd kunnen worden. Dit prototype ondersteunt de ontwerper om een machine sneller, eenvoudiger, en met minder fouten te ontwerpen.

De aanbevelingen voor verder onderzoek zijn de volgende. Het prototype moet verder ontwikkeld worden zodat een hulpmiddel wordt verkregen dat geëvalueerd kan worden in de praktijk. Diverse modellen in het meta productmodel kunnen verbeterd worden en aanvullende aspectmodellen kunnen toegevoegd worden. Nieuwe ontwerpactiviteiten kunnen worden gedefinieerd en worden toegevoegd aan de huidige activiteiten. De database moet gevuld
worden met een aanzienlijke hoeveelheid informatie zodat de ontwerper meer alternatieven vindt. Verder zal onderzocht moeten worden hoe het prototype aansluit bij commerciële pakketen en hoe de ontwerpfasen voor en na de conceptuele ontwerpfase in het prototype geïntegreerd kunnen worden.
Contents

Acknowledgements .............................................. v
Summary .................................................................. vii
Samenvatting ........................................................... ix

1 Introduction .......................................................... 1
   1.1 Trends in industry ............................................. 1
   1.2 Initial problem statement ................................. 2
   1.3 Goal of the research ....................................... 3
   1.4 Scope of the research .................................... 3
       1.4.1 Background ........................................... 3
       1.4.2 Application domain: production machines .... 3
       1.4.3 Conceptual design .................................. 4
   1.5 Hypothesis ....................................................... 6
   1.6 Research approach ....................................... 7
   1.7 Terminology ................................................... 7
   1.8 Structure of the thesis ................................... 8

2 State of the Art .................................................. 9
   2.1 Introduction .................................................. 9
   2.2 Models of the design process ......................... 9
       2.2.1 Prescriptive models ............................... 10
       2.2.2 Descriptive models ............................... 11
       2.2.3 Discussion ........................................... 14
   2.3 Product models ............................................. 15
       2.3.1 Theory of technical systems ................... 15
       2.3.2 Theory of domains ............................... 16
       2.3.3 Function-means tree .............................. 17
       2.3.4 Function-Behaviour-State modelling ......... 18
       2.3.5 Specific considerations ......................... 19
           2.3.5.1 Models with multiple views ............... 19
           2.3.5.2 Designing with functions ................. 21
           2.3.5.3 Decomposition ............................... 22
       2.3.6 Discussion ........................................... 24
4.3.3 Data schemas of the aspect models................................................. 72
4.3.4 Data schema of the meta product model........................................... 74
4.3.5 Mapping rules.................................................................................... 76
  4.3.5.1 Mapping with the product state model ......................................... 77
  4.3.5.2 Mapping with the function model............................................... 78
  4.3.5.3 Mapping with the solution model............................................... 79
  4.3.5.4 Mapping with the solution layout model...................................... 79
  4.3.5.5 Mapping with the state transition model..................................... 79

4.4 Conclusions ....................................................................................... 81

5 Procedures of Design Activities .............................................................. 83

5.1 Introduction ......................................................................................... 83
  5.1.1 Overview ....................................................................................... 83
  5.1.2 Analyses of design activities ......................................................... 84
  5.1.3 Direction for improvements ......................................................... 84

5.2 General design activities ..................................................................... 86
  5.2.1 Administration activities .............................................................. 86
  5.2.2 Presentation activities ................................................................... 89
  5.2.3 Synthesis activities ....................................................................... 90
  5.2.4 Analysis activities ......................................................................... 91
  5.2.5 Evaluation activities ...................................................................... 91
  5.2.6 Design activities initiating other design activities ......................... 91

5.3 Design activities for production machines ......................................... 92
  5.3.1 Administration activities .............................................................. 92
  5.3.2 Presentation activities ................................................................... 95
  5.3.3 Synthesis activities ....................................................................... 96
  5.3.4 Analysis activities ......................................................................... 99
  5.3.5 Evaluation activities ..................................................................... 100

5.4 Conclusions ....................................................................................... 101

6 Data schema of Background Information .............................................. 103

6.1 Introduction ......................................................................................... 103
  6.1.1 Overview ....................................................................................... 103
  6.1.2 Analysis of current databases ....................................................... 104
  6.1.3 Direction for improvements ........................................................... 105

6.2 General data schema .......................................................................... 106
  6.2.1 The three-level ANSI-SPARC Architecture .................................... 106
  6.2.2 Conceptual data schema ............................................................... 107
  6.2.3 Logical data schema ..................................................................... 109
  6.2.4 Transactions .................................................................................. 110

6.3 Data schema for production machines .............................................. 111
  6.3.1 Conceptual data schema ............................................................... 111
  6.3.2 Logical data schema ..................................................................... 115
  6.3.3 Transactions .................................................................................. 117

6.4 Conclusions ....................................................................................... 118
# Table of Contents

7 Implementation and Case Studies ........................................... 121
    7.1 Introduction........................................................................ 121
    7.2 Prototype design support system .................................. 121
    7.3 Case studies....................................................................... 124
        7.3.1 Case 1: Block-packaging .................................. 124
        7.3.2 Case 2: Meat-taper ........................................ 127
        7.3.3 Case 3: Flow-packer .................................... 130
        7.3.4 Discussion.................................................... 131
    7.4 Desired improvements.................................................. 132
        7.4.1 Product model.............................................. 132
        7.4.2 Design activities......................................... 136
        7.4.3 Database.................................................... 138
    7.5 Conclusions..................................................................... 138

8 Conclusions ............................................................................ 141
    8.1 Introduction...................................................................... 141
    8.2 Review of the research goals ....................................... 141
    8.3 Deliverables..................................................................... 143
        8.3.1 Model of the design process .......................... 143
        8.3.2 Meta product model .................................... 143
        8.3.3 Procedures of design activities .................... 144
        8.3.4 Data schema ............................................... 144
        8.3.5 Prototype design support system ................. 145
    8.4 Recommendations for future research ......................... 145

Bibliography .............................................................................. 147

Appendix A Formal Description ............................................. 155
    A.1 Glossary ...................................................................... 155
    A.2 Data schemas of the aspect models.......................... 156
    A.3 Data schema of the meta product model .................. 160
    A.4 Mapping rules.......................................................... 162

Appendix B Prototype Design support system ...................... 165
    B.1 Main window............................................................ 165
    B.2 Views ...................................................................... 166
    B.3 Browsing the database ............................................ 169
    B.4 Searching the database ........................................... 170

Curriculum Vitae ...................................................................... 171
Introduction

This chapter gives an overview of the problem domain treated in this research. First, general developments within industry are mentioned which initiate the need for design support tools. An initial problem statement is presented, followed by the goal of the research project. Further, the scope, hypothesis, and approach of the research are given. The chapter ends with an explanation of the terms used and an outline of the contents of this thesis.

1.1 Trends in industry

A number of developments within industry make it necessary to reconsider the design process.

First, there is the tendency to shift from producers' market to a consumers' market. Due to this, companies have to reduce costs and time-to-markets. Product costs are for about 80% determined during design, while about 20% of the costs are made (Eversheim, 1990). Therefore, design will play a crucial role in reducing product costs and the time-to-market.

Second, the requirements that products must meet are changing: new requirements have been introduced and existing have been tightened. Besides reducing the cost and increasing the quality, the product must meet aesthetic and liability requirements as well.

Third, the complexity of products has increased. A method to cope with complexity is to divide the product into smaller chunks, which are designed by individual designers within a team of designers. Furthermore, designers must consider knowledge from several disciplines, e.g. mechanical, electrical, and software engineering. Because no designer can overview all these domains, it
will lead to specialisation in practice and thus for the need of teamwork and communication (Blessing, 1994 & 1996).

Stimulated by these developments, both industry and research institutes are paying more attention to the design process. This resulted in a number of models, methods, and computer-tools that describe, prescribe, and/or support the design process. However, improvements are required to keep up with the demanding market.

1.2 Initial problem statement

The conceptual design stage is hardly supported by tools, especially when the design of production machines is regarded. Designers are often thrown on their own resources while they document designs, perform activities, and reuse information from past designs.

During the early stages of design, conceptual designs are made that form the basis for the remaining design stages. Conceptual designs can contain geometric information, but also non-geometric information like realised functions, fulfilled requirements, and simulation results. All these types of information should be documented, but in practice, mainly geometric information is stored. One of the reasons is that there are no strict rules or standards that prescribe how non-geometric information should be documented.

The designer generates the design by performing all kind of activities. The design is modified, analysed, evaluated, simulated, and so on. Some of these activities require the creativity, experience, and knowledge of the designer, while others are very repetitive, monotonous, and time consuming. If these activities are supported or even automated, designers can spent more time on the creative activities.

Past designs can contain information that can be reused in new design cases. However, before past designs can be reused, it must be known how they can be documented. The information about past designs must be processed and filtered so that it can be applied in new designs.

Companies and research institutes feel that support of the conceptual design stage is possible and necessary. Therefore, tools must be developed that support the conceptual design stage. To achieve this, it must be investigated how designs can be described, created, and reused, and how a design support system can assist designers during these tasks.
1.3 Goal of the research

The main goal of this research is:

To investigate how designers of production machines can be supported during the conceptual stage of the design process.

The following objectives are identified to achieve this:

- To review existing design process models, product models, design theories, and computer-based approaches for supporting the design process in order to obtain insight in the design process and to identify the limitations and opportunities of the approaches.
- To investigate how designers can be supported during the conceptual design stage.
- To develop a computer-based approach to support the conceptual design process and
- To evaluate the approach by implementing it into a working prototype and determine its feasibility and effectiveness.

1.4 Scope of the research

1.4.1 Background

Since the 70's, the laboratory of Mechanisation of Production, Delft University of Technology, is involved in the research project CADOM (Computer Aided Design Of Mechanisms). The project aims at the synthesis of mechanisms [Rankers et al., 1976], analysis of mechanisms [van der Werff, 1977], optimisation of mechanisms [Klein Breteler, 1987] and the computer aided design of mechanical systems [Zhang, 1994]. The project resulted in several computer-tools that focus on mechanisms and on the analysis of designs. More research is required that focuses on production machines and on the synthesis or conceptual design of these machines.

1.4.2 Application domain: production machines

This research focuses on the design of dedicated production machines for discrete products. Production machines produce products of similar shape and quality in large amounts with high production rates. Examples are packaging machines for cookies, bottling-machines and machines to produce TL-lamps (Figure 1).
Production machines process the incoming materials in several stages and transform them into the required product. Several units execute the transformation processes. Between these units, the materials are transported and positioned. Control systems ensure the correct quality and course of the production process.

1.4.3 Conceptual design
The research focuses on conceptual design. The conceptual design phase is the most important phase of the design process since in this phase decisions are made which determine the course of the remaining design process and later processes, e.g. assembly and production.

To position conceptual design within the design process, the innovation-process as described by Roozenburg & Eekels (1995) is used (Figure 2). It starts with the formulation of company goals and strategies. This is the input for the generation and selection of ideas for possible products. Selected ideas are used for further development and transformed into an actual realisation. Roozenburg & Eekels define strict development as the transformation of the chosen product ideas into finished designs. The strict development stage is divided into production development, product design, and marketing planning.
Conceptual design falls into the domain of strict development and *product design* in particular. Throughout this thesis, the term design process indicates product design.

To place conceptual design within the design process, the model of the design process as defined by Pahl & Beitz (1993) is used. They divide this design process into several stages (Figure 3). During the *clarification of the task*, needs are analysed and problems, goals, and requirements are formulated. In the *conceptual design stage*, solution-principles or concepts for the product and its main elements are generated based on the functions the product has to fulfil. Furhter, the sub-solutions are synthesised into an overall solution. During *embodiment design*, an initial physical design including component arrangements and initial forms is designed. In the *detail design stage*, the final physical design, which includes drawings, manufacturing and assembly instructions, and so on, is created.
1.5 Hypothesis

The hypothesis used throughout this thesis is that the (conceptual) design process can be modelled by three elements: a product model, design activities, and a database (Figure 4).

The product model contains all documented information about a design as it exists at a certain point in time. A design activity is an operation that results in an extension or modification of the database and/or product model. The database contains background information about designs that can be reused in new designs, or more accurate, in product models of new designs.
Figure 5 shows how a product model is incrementally created in time due to interactions with design activities. A product model as it exists at time $i$ (PM$_i$) is modified by a design activity (DA$_i$), which results in a new product model at time $i+1$ (PM$_{i+1}$). In similar manner, the database can evolve in time. The chain of design activities can be seen as the design process.

![Figure 5 Evolving product model and executed design activities](image)

This hypothesis is elaborated more in chapter 3.

### 1.6 Research approach

The research started with a literature survey of design process models, product models, design theories, and computer-based approaches that can contribute to the development of a product model, design activities, and database for production machines. Suitable models, methods, and approaches were synthesised into a model of the design process of production machines. From time to time, designers were consulted to verify and judge the developed models. Since the models are very abstract and academic, designers found it hard to comment on them. Therefore, the models were made more concrete and practical by implementing them in a design support system. This prototype design support system was used to perform several case-studies to determine the feasibility of the models.

### 1.7 Terminology

This section contains a summary of the most important terms used in this thesis. The presented definitions will be used throughout the thesis.

**Product model**

The product model contains all documented data about a design as it exists at a certain point in time.

**Meta product model**

The meta product model defines what data is documented in the product model, the relationships among the data, the constraints on the data, and the semantic
information about the data. The meta product model can be seen as a model of the product model.

**Design activity**
A design activity is an operation that results in an extension or modification of the product model, database, or both.

**Procedure of a design activity**
A design activity procedure prescribes how a design activity is performed, which input it requires and how it affects the product model, database, or both.

**Database**
A database contains background information about designs that can be reused in the product model of new designs. It is a collection of logically related data (and a description of this data), designed to meet the information needs of an organisation (Connolly et al., 1999).

**Data schema**
A data schema describes what data is stored in the database, the relationships among the data, the constraints on the data, and the semantic information about data.

### 1.8 Structure of the thesis

Chapter 2 reviews the current state of the art of design models, methods and tools in order to identify opportunities and possible shortcomings of these approaches. Chapter 3 refines the problem description, describes the hypothesis in more detail, and elaborates on the pursued design support system. The hypothesis consists of three parts: a **product model** that describes the production machine, **design activities** that create the design, and a **database** that contains information that can be reused. Chapters 4, 5, and 6 describe these elements in detail. Chapter 7 elaborates on the implementation of the models and presents the results of case-studies performed with this implementation. Chapter 8 concludes this thesis and gives recommendations for future research.
2

State of the Art

2.1 Introduction

Much research has been performed to support the design process. This resulted in numerous methods, models, and computer-tools. Although this chapter reviews a considerable amount of models, it does not have the intention to be complete. The aim is to obtain insight in the design process and to determine which methods and models can be used to develop a product model, design activities, and database for the design of production machines.

The chapter uses the hypothesis to categorise the discussed models. First, several models of the design process are described to determine whether they reflect the hypothesis. After that, three sections elaborate on the particular elements of the hypothesis and review product models, design activities, and databases. Subsequent section discusses software tools that support the design process. In case a model, method, or tool contains more than one element of the hypothesis, it is placed in the category that fits most.

The chapter concludes with shortcomings and opportunities of the models, methods, and tools.

2.2 Models of the design process

This section describes models of the design process. The aim is to get insight in the design process and to determine whether a product model, design activity, and database can be recognised in these descriptions. The models are categorised in prescriptive and descriptive design process models (Blessing, 1994). Prescriptive models define how the design process should be; descriptive models describe how the design process actually occurs. The section ends with a discussion about the models design process models.
2.2.1 Prescriptive models
Methodical Design concerns a range of prescriptive models of the design process and originates from German literature. Most of these models describe the design process by dividing it into several stages. Another characteristic is that the methods often use the functions of the product as a means to design the product.

Pahl & Beitz (1993) and the VDI (Verein der Deutsche Ingenieure) define most well known methodical design approaches. Figure 3 already showed a simplified version of the Methodical Design method of Pahl & Beitz. Figure 6 shows the General Approach to Design as formulated by the VDI (1986). The design process is divided into several steps that must be executed in predetermined order. The phases within the VDI-method correspond roughly to the four steps defined by Pahl & Beitz.

Figure 6 General Approach to Design (VDI 2221, 1986)
The VDI (VDI 2222, 1977) developed a prescriptive model of the conceptual design stage (Figure 7). The conceptual design stage starts with the overall function that the product has to fulfil. The overall function is decomposed into several sub-functions to divide the complex design problem into smaller and easier problems. Sub-solutions that can realise these sub-functions must be designed and are combined into several overall solutions. These solutions result in different concepts.

![Diagram of conceptual design stage](image)

**Figure 7** Conceptual design stage according to the VDI 2222 (1977)

### 2.2.2 Descriptive models

Roozenburg & Eekels (1995) define the basic cycle of design as depicted in Figure 8. The underlying thought is that design is a trial and error process, existing of sequential empirical cycles, in which the knowledge about the problem and the product to be designed increases.
Gero (1990) formulated a generic, descriptive model for design (Figure 9). In his view, design is a transformation of a function $F$ into a design description $D$. However, since such a direct transformation does not exist, behaviours are introduced. In his model, the design description represents the elements of the product and their relations as denoted by the structure $S$.

\[
\begin{align*}
F & \quad \rightarrow \quad S \quad \rightarrow \quad D \\
\downarrow & \quad \downarrow & \quad \downarrow \\
B_s & \quad \longleftrightarrow \quad B_s
\end{align*}
\]

$F = \text{Set of Functions}$

$S = \text{Structure}$

$D = \text{Design Description}$

$B_s = \text{Set of Expected Behaviours}$

$B_s = \text{Set of Actual Behaviours}$

\[\text{-----} = \text{Transformation} \quad \text{-----} = \text{Occasional Transformation} \quad \longleftrightarrow = \text{Comparison}\]

**Figure 9** A generic model for design (Gero, 1990)
Figure 10 shows that the generic model of design distinguishes several phases.

![Diagram showing seven phases of design](image)

**Figure 10** Phases in the generic model of design according to Gero (1990)

Ullman et al. (1988) describes the TEA (Task-Episode Accumulation) model. A detailed analysis of designers during the design process is the basis for this model. The fundamental components of the model are the design state, design operators, episodes, and tasks. The design state contains information about the evolving design, including problem specifications, additional constraints introduced by the designer, proposed designs, drawings, calculations and so on. Design operators are primitive information processes that modify the design state by performing calculations and simulations, creating new proposed designs, evaluating proposed designs, and making decisions. The TEA-model contains 10 operators: select, create, simulate, calculate, compare, accept, reject, suspend, patch, and refine. To accomplish a design, the designer applies the primitive operators in meaningful sequences called episodes. The TEA model has six episodes: assimilate, document, plan, repair, specify, and verify. A task is a collection of related primitive goals. Following tasks are defined: conceptual design, layout design, detail design, and catalog selection.

Suh (1990) gives a more abstract description of (conceptual) design. This description, called axiomatic design, defines Functional Requirements (FRs) and Design Parameters (DPs). The Functional Requirements form the minimum set of independent requirements that completely characterise the functional needs of the product design. The Design Parameters are the essential variables that characterise the physical entity created by the design process to fulfil the FRs. Design is thus the mapping between of the functional domain to the physical domain (Figure 11). The functional decomposition evolves in parallel with synthesis of the technical system and is documented into one structure for the FRs and one for the DPs.
Suh defines two design axioms as guidance for the design process:

Axiom 1: The Independence Axiom
Maintain the independence of functional requirements

Axiom 2: The Information Axiom
Minimise the information content

The first axiom distinguishes between good and bad design. The second axiom is the criterion for selection of optimum design solutions from among those that satisfy axiom 1.

2.2.3 Discussion
The models describe or prescribe how the design process takes place. Although based on design experience, the models have a high level of abstraction and describe the design process almost on "management" level. Due to this abstraction level, the models do not directly support the designer. However, they give him a guide on how to continue and finish the complete process. For the development of design support systems, the models can give more understanding, but a direct implementation is often not possible. The models must be formalised in a manner suitable for implementation and specified for a certain application domain.

The models consist of number of design activities, often supplemented with the sequence in which they should be carried out. Further, they show the resulting product models of these activities, like the function structure or a preliminary design. Although the activities give a good overview of what has to be done during design, they do not indicate how these activities should be performed. Similar, the results of these activities show what must be realised, but they do not indicate how these results can be described in detail. A description and the usage of a database is absent in all models.
The several design activities can be used to determine the type of required design activities for the design of production machines. Specifications, functions, and principle solutions can be useful to model a production machine.

2.3 Product models

This section reviews product models that contain suitable elements for developing a product model of production machines. Besides describing the models, a number of specific considerations about product models are made: designing with functions, multiple views, and decomposition. The section closes with a discussion about the product models.

2.3.1 Theory of technical systems

Hubka & Eder (1988) developed the Theory of Technical Systems (TTS), which mainly concerns transformation processes. Figure 12 shows the model.

![Figure 12 General model of the transformation process (Hubka & Eder, 1988)](image)

The model builds upon, amongst others, following elements and terms, which can be useful to describe a production machine:

- **Transformation (Tr):** changes certain properties of the operand (as passive member of the transformation process), by mutual interaction between object and means
- **Operand (Od):** What is being transformed. Object that is being changed in the transformation from an input state to an output state
- **State:** Aggregate (vector) of values of properties (Pr) of a system at a certain time
• Operators (Op): Who and what delivers the necessary effects (as active members) to the operands
• Time: When is it being transformed? Time period during which the transformation occurs.

2.3.2 Theory of domains
The basis of the Theory of Domains (Andreasen, 1980 & 1992) is that a limited number of domains can describe a product:

• process domain: a structure of processes describes the product. A process is a transformation of the operand material, energy, and/or signals. A process is based on a technology;

• function domain: a structure of functions describes the product. A function is the ability to create an effect. In contrast to a process, a function does not describe a transformation;

• organ domain: a structure of organs describes the product. Organs are the function carriers; i.e. functions are realised by organs. The organs are materialised surfaces, having mechanical relations;

• component domain: a structure of components describes the product, i.e. parts and groups of parts. Components are the carriers of organs.

Between the domains, causal relations exist, as shown in Figure 13. Together, the domains form the product chromosome.

Figure 13 The chromosome of a product (Andreasen, 1992)
Hansen (1997) states that there are three types of relations between organs in the chromosome model:

- coupling, where the output from organ A is the input for organ B,
- time relation, where A+B is controlled to create the desired functionality,
- arrangement, where the spatial arrangement of A and B realises functionality.

### 2.3.3 Function-means tree

The Function-Means Tree (Tjalve, 1978) shows the functions as well as possible means to realise these functions. It is based on the observation that functional decomposition cannot be performed unless at least some knowledge is available about the realisation of the function. Figure 14 gives an example of the Function-Means Tree and shows alternative solutions for a certain function. The grey boxes show chosen alternative means.

![Function-Means Tree](image)

**Figure 14 The Function-Means Tree (Andreasen, 1980)**

Svendsen & Hansen (1993) give following design procedure to be used with the Function-Means Tree:

- describe purpose functions
- search means with configuration that realise the function
- verify that the means are at the same abstraction level
- choose "the best" means between the alternative means
- adjust the configurations on all levels above the chosen means level
- if the type of means is not a functional surface then set up the additional function for the means, else stop

The procedure ends up with functional surfaces at the bottom of the Function-Means Tree, which afterwards have to be integrated to components.
2.3.4 Function-Behaviour-State modelling

The Function-Behaviour-State (FBS) model (Umeda et al., 1990) represents a state of a design object by entities, attributes of the entities and relations among entities. Behaviour is defined as "sequential changes of states". The set of states and behaviours is called viewpoint. A designer has various kinds of viewpoints.

A function is a description of behaviour abstracted through recognition of behaviour for utilisation. Behaviour is an objective concept grounded on states and described within a viewpoint; a function is a subjective concept related to behaviours. Figure 15 shows the relationship between function, behaviour, and state.

![Figure 15 Relationship among function, behaviour and state (Umeda et al., 1990)](image)

In Umeda et al. (1991, 1992), the FBS-modeller is used to design a photocopier by using function redundancy, which means that potential functions of existing parts are used. Tomiyama et al. (1993) extend the FBS-modeller with a CSP (Control Sequence Program), which means that states can be arranged in time.

Iwata & Onosato (1989) describe a similar modelling technique. They define a state-effect formalism to describe the operations within machines. A state is the representation of a design object and consists of a spatial region, a time interval, and a set of properties. An effect is defined as the physical influence that causes a state transition or holds a state that makes a spontaneous transition. A set of states represents an object and state transitions represent the behaviour of the object. This modelling technique incorporates the changes of the machine in operation.
2.3.5 Specific considerations
The product models have similar elements. First, they often describe the product by using multiple views or models. Second, they describe the functions that the product should fulfil. Third, they use decomposition to divide the overall problem into smaller and easier problems. This section elaborates on these similar elements.

2.3.5.1 Models with multiple views
During design, a designer simultaneously uses several models of the product to be designed. Examples are the function- and solution-structure. Each of these models describes a specific view on the product. Previous sections already mentioned a number of models that use multiple views. Axiomatic design (section 2.2.2) defines two spaces to model the product: a functional and physical space. The chromosome model (section 2.3.2) distinguishes four domains: the process, function, organ, and component domain. The FBS-model (section 2.3.4) consists of three views: a function, behaviour, and state view. This section presents several other product models that use multiple views or multiple models to describe a product.

Erens & Verhulst (1995) describe three domains (Figure 16):
- The functional domain is a consistent description of the functionality of a system and strongly relates to the purpose of the product.
- The technology domain is a consistent description of the application of technologies to ensure the operation of the product.
- The physical domain is a consistent description of the physical realisation of a system and strongly relates to the construction of the product.

![Figure 16 Domains to describe a product (Erens & Verhulst, 1995)](image)

Kiriyama et al. (1989) and Yoshioka & Tomyama (1997) describe a metamodel mechanism, which integrates and maintains consistency among various models. Figure 17 shows the principle of the metamodel. An aspect
model is a model of the design object from a particular point of view, such as deformation, shape, and motion.

Figure 17 The metamodell mechanism (Yoshioka & Tomiyama, 1997)

Kjellberg & Schmekel (1992) state that a designer needs different product models during the product development process. Besides the variety of models, the relations between the models should remain consistent. In contradiction to Yoshioka & Tomiyama, they define dependency relations between the models to achieve this (Figure 18).

Figure 18 Models connected by dependency relations (Kjellberg & Schmekel, 1992)
2.3.5.2 Designing with functions

Product models often specify the functionalities that the product must fulfil. Previous sections already mentioned a number of models that do this. This section elaborates more on designing with functions.

Umeda et al. (1996) states that the representation and manipulation of functions are crucial issues for constructing a design system that supports conceptual design. Hashemian & Gu (1997) have the opinion that the formal representation of functions is one of the essential factors for developing computerised conceptual design tools. They describe a function with actions and operands, which are defined by their type and attributes. The usage of taxonomies (Figure 19) assists during the creation of actions and operands.

![Diagram](image)

**Figure 19** Elements of a function in the proposed scheme by Hashemian & Gu (1997)

Hashim et al. (1994) state that there are many difficulties in representing functions. If progress is to be made in the development of intelligent CAD systems, functions must be explicitly represented in a form both human and computer can comprehend. He defines a representation as depicted in Figure 20, where a function connects a functor (the part that provides the functions) to an acceptor (the part that accepts the functions). The function can be selected from a list with standard primitive functions. The method is used for redesigning products. An algorithm is defined to structure an existing design according to the presented approach.

![Diagram](image)

**Figure 20** Conceptual representation of Functor, Function and Acceptor

21
Ullman (1993) mentions several limitations of designing with functions. One of them is the lack of variation of functions during different operational phases, which is the usual case for most products. It is emphasised that the relations between functions are very important. Chakrabarti (1997) states that function-structures did not give deep understanding and problem solving. He concludes that the abstraction of functions encompasses the solutions, instead of indicating them. Simultaneous consideration of both functions and means that fulfil them is a solution for these problems. Kota & Ward (1990) argue that functions are elegant because of the attractive simplicity of the functional designation. However, the detail of functions is too course to support general mechanical design, e.g. a chair performs the function support, but so do bridge stringers, a bed, and table legs. However, in electric- and hydraulic-design, functional design has been proven to fit automated design.

**2.3.5.3 Decomposition**

Decomposition is a method to divide complex problems into smaller, easier ones. It refers mostly to the decomposition of functions. Erens, McKay & Bloor (1993) state that besides functional decomposition, physical decomposition is also very important. Besides that, the consistency between these decompositions should be maintained.

Function-oriented decomposition makes it possible to choose between alternative solutions on all levels. Other reasons for functional decomposition are (Ringstadt, 1997):

- ensure systematics in design to cover the solution space;
- focus on the purpose of the design, instead on the design itself;
- support documentation of the design process and the fundamentals of the design;
- facilitate changes in the design without losing other purposes of the design.
Pahl & Beitz (1993) define functional-decomposition as shown in Figure 21. The in- and outputs of functions are material, signal, and energy.

![Figure 21 Decomposing functions (Pahl & Beitz, 1993)](image)

Umeda et al. (1996) distinguish two types of functional decomposition: causal and task decomposition. When causal decomposition is considered, the sub-functions are causally related. In task decomposition, the sub-functions occur independently. Shimomura et al. (1995) describe following relations between functions: decomposed-into, conditioned-by, and enhanced-by. Decomposed-into indicates that a function is decomposed into sub-functions. Conditioned-by denotes that a new function is needed to actualise a certain function. Enhanced-by denotes that a new function is not necessarily needed, but that it enhances the functioning of another function.

Ringstadt (1997) compares two approaches for functional decomposition: the function-means-tree (section 2.3.3) and the axiomatic approach as defined by Suh (section 2.2.2) and concluded that both methods are very similar. The difference is that Suh uses the functional decomposition to analyse the solutions, whereas the function-means-tree uses decomposition as a means to support the synthesis of solutions.

Kusiak & Szczepicki (1990) describe a formal method to decompose design specifications and functions by using logical trees to generate different conceptual solutions. The underlying thought is that requirements can be decomposed in different ways and that each sub-requirement can have different functions. The solutions are systematically generated until each function is fulfilled.
Kusiak, Szczerbicki & Park (1991) state that little literature has been published on the application of multiple levels of abstraction in design, although the idea appears to be useful to find an interesting, innovative solution. According to Mortensen (1993) there are no systems, that can describe functional decomposition of goal specifications into subsystems on a theoretical basis.

Erens & Verhulst (1995) define the role decomposition and composition in correlation to other design steps (Figure 22):
- Decomposition is adding detail to a product model;
- Allocation is the creation of relationships between elements of different product models;
- Composition is combining elements of a product model;
- Validation is a check on the realised quality level of the product model, by relating it to a previous product model.

![Functional Model, Technology Model](image)

**Figure 22** Four elementary design steps (Erens & Verhulst, 1995)

Hashemian & Gu (1997) give three rules for decomposition. An example of such a rule is: if N solutions can provide one function, decompose the function in N sub-functions that correspond to one of the available solutions.

**2.3.6 Discussion**
Product models are often very general since they were developed to describe a wide range of products. Consequently, specific properties of a product, like the motions of the components in a production machine, are often not modelled. Besides, the used terminology in the models is often very general and academic and does not correspond to the terminology that designers use.

A production machine consists of several processes that transform the incoming materials. Therefore, the theory of technical systems contains suitable elements
to model production machine since it describes transformation processes and the operands that are being transformed. The function-means-tree is useful because it shows the functions that a means requires to realise its function and the alternative solutions for a function.

The principle to describe a product by using multiple views connects highly to the designers' way of thinking. Therefore, it should be incorporated into the model of production machines. A meta model or dependency relations can be applied to maintain the consistency between the views. Views describing the functions and solutions can be suitable to model production machines. The mentioned function taxonomies can serve as basis to develop a database with functions. Besides these general views, views that show specific properties of a production machine should be developed and incorporated in the product model of production machines.

Decomposition is a powerful method to simplify the design problem. It should be defined for all views. Besides decomposition, its counterpart composition must be available. However, a formal definition of decomposition and composition is lacking in many models.

2.4 Design activities

This section elaborates on design activities that a designer performs during the design process. The aim is to determine which activities can be useful to design production machines and how they can be modelled. The activities mentioned in the design process models are reviewed and there is elaborated more on specific activities like generating ideas, configuration design, and reasoning. The section ends with a discussion about the activities.

2.4.1 Design activities in the design process models

The models of the design process (section 2.2) contained a number of design activities. Table 1 presents these activities and places similar activities in rows. From the overview, two observations can be made. First, design activities can have several levels of detail. For example, the synthesise activity contains or correlates to several more detailed and smaller activities, e.g. determine, search, divide, and develop. Second, design activities only have a clear meaning when the part of the product model that they affect and/or use is known. For example, the activity "create" only makes sense when it is known what must be created.
2.4.2 Generating ideas

Many methods to generate solutions are described in literature. Cross (1989) summarises following methods: Brainstorming, Synectics, and Removing Mental Blocks. Roozenburg & Eekels (1995) mention Brainwriting, Method 6-3-5, and Structured Free Association. A characteristic of these methods is that they are mainly group exercises to generate ideas in an informal matter, which makes it very difficult to support these method by computer-tools.

A technique to create innovative en creative solutions in a systematic way is the morphological chart. Morphological charts (Zwicky, 1966) are a means to find all theoretical solutions for a problem. Another characteristic is the rigid separation of synthesising and selecting solutions. The chart gives an overview of the functions and the synthesised solutions. The method works roughly as follows: first, all the required sub-functions must be defined and subsequently, solutions must be generated for each of these functions. The functions and solutions are placed in a morphological chart (Figure 23).
A total-solution is found by selecting one solution for each function and combining them into one overall solution, as shown in Figure 23 by lines. During this composition, the compatibility between the selected solutions must be checked.

2.4.3 Configuration design

Configuration design is a type of conceptual design activity in which physical systems are synthesised from a set of predefined components that can only be combined in certain ways (Kota & Lee, 1993-1). Central to this definition are two central tasks: (1) selection of appropriate components and (2) determination of their interconnections.

Yu & MacCallum (1997) give following definition of configuration: "Configuration is the process of creating an arrangement from a given set of elements by defining the relationships between selected elements such that the arrangement satisfies the design requirements and design constraints. Configuration consists of configuration design which is the design activity of creating configuration solutions, and configuration management, which is the process of maintaining a consistent configuration under change."
2.4.4 Reasoning

Case-Based Reasoning
A case-based reasoner solves new problems by adapting solutions of previously solved similar problems to the new requirements. It is built upon the premise that human beings reason mostly from experience and not from first principles. A case-based reasoning system (Figure 24):

- finds and retrieves solution from it case-base that have met the same or similar requirements
- adapts the retrieved solutions to meet the current requirement
- evaluates and if necessary repairs the adapted solution
- suggests the solution it found to the user and
- learns new cases from the solution it generates (Göker & Birkhofer, 1994).

![Figure 24 A case-based reasoning flowchart](image)

Rule-Based Reasoning
A rule-based reasoner uses rules about a product or a process in order to support design. An example of a rule for water-jet cutting is:

\[
\text{IF material thickness } \geq \text{... mm THEN pressure } \geq \text{... bar}
\]

Constraint-Based Reasoning
During constraint based reasoning, changes made in a design are immediately checked against constraints. An example is that when an axis must fit in a hole, the size of the axis can only be altered within certain constraints, namely the size of the hole.
2.4.5 Discussion
The design process models show that design activities can have several levels of detail and that the part of the design model that the activity affects must be known. For example, "select" is not as comprehensive as "select function".

The methods to generate ideas are mainly group exercises and therefore hard to support by computer. Morphological charts are often applied in practice. Originally, the method was meant to generate all possible solutions to a problem. However, in practice the method is mainly used to represent possibilities and to abstract the design problem. Nevertheless, it can be used to support designers of production machines.

Configuration design is especially useful when it is known which components a product must have and what alternative components are. However, production machines are too versatile to be configured in this manner. Checking the interconnection between components can be useful to determine the interconnection of the sub-solutions in a production machine.

Case-based-reasoning can only be effective when a model of the cases exists; in other words, a product model of production machines must be available. A similar argumentation can be held for constraint based reasoning. Rule based reasoning is only effective when proper rules are available and it must be determined which rules can support the conceptual design of production machines.

2.5 Databases
This section elaborates on databases that can be used to support the design of production machines. Methods to develop and describe a database will not be reviewed. The theory of inventive problem solving is explained, followed by function taxonomies and catalog design. The section ends with a discussion about the databases.

2.5.1 Theory of Inventive Problem Solving
The Theory of Inventive Problem Solving (TIPS) was developed by Altshuller (1988) with very little knowledge about other methodologies and is based on a large empirical knowledge base in the form of patents. No other methodology is based on that kind of empirical base (Malmqvist et al., 1996).

Altshuller analysed a large number of patents (> 2 million) and found general patterns. The analyses showed that most patents describe solutions to solve conflicts within a system. A conflict means that when one design parameter is
improved, it causes one other to deteriorate. Furthermore, similar underlying principles are the basis for many solutions. In total, about 40 principles were found.

TIPS consists of three subsystems (Fey, Rivin & Vertkin, 1994):
- the Algorithm for Inventive Problem Solving,
- Standard Solutions to Inventive Problems and
- the Database of Physical, Chemical and Geometrical Effects.

Malmqvist et al. (1996) compares TIPS with the methodical design of Pahl & Beitz (1993) and gives an integration of the methods, focusing on the clarification of the tasks and the conceptual design stage.

### 2.5.2 Function taxonomies

Since functions play a very important role in the conceptual design stage, many attempts have been made to create a taxonomy of functions. Usually, a verb and a noun (e.g. "increase torque") describe a function. A human designer can state functions whatever terminology he prefers, but a computational design system must rely on a limited number of elementary functions (Malmqvist, 1994). Krumhauer (1974), Koller (1976), Rodenacker (1970), Roth (1982), and Kirschman et al. (1996) describe examples of function-taxonomies. Table 2 shows an example of primitive functions as defined by Hashim et al. (1994).

<table>
<thead>
<tr>
<th>Key Functions</th>
<th>Synonymous Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support</td>
<td>Carry, Mount</td>
</tr>
<tr>
<td>Hold</td>
<td>Fasten, Join, Fix, Attach</td>
</tr>
<tr>
<td>Locate</td>
<td>Position</td>
</tr>
<tr>
<td>Constrain</td>
<td>Limit, Restrain, Restrict</td>
</tr>
<tr>
<td>Seal</td>
<td>Contain, Store</td>
</tr>
<tr>
<td>Cover</td>
<td>House, Shield</td>
</tr>
<tr>
<td>Drive</td>
<td>Push, Transmit, Force</td>
</tr>
<tr>
<td>Guide</td>
<td>-</td>
</tr>
<tr>
<td>Clamp</td>
<td>Grasp, Grip, Secure</td>
</tr>
<tr>
<td>Couple</td>
<td>Connect, Link</td>
</tr>
</tbody>
</table>

*Table 2* Primitive functions (Hashim et al., 1994)

Hundal (1990) argues that it is more advantageous to work with basic or generally valid functions instead of task-specific functions. He distinguishes six basic functions: channel, change magnitude, store/supply, connect, branch and convert. The majority of task-specific functions can be expressed in terms of one or a combination of these basic functions. Hundal & Byrne (1990) describe a flow-chart to determine the basic function.
2.5.3 Catalog design
Catalog Design, often described within German literature, describes how catalogues with solutions can be used during design. Kota & Lee (1993-1) organise the design knowledge as depicted in Figure 25. During design, alternatives can be obtained by (1) enumerating alternative topologies of the same set of functions, (2) choosing alternative generic devices and (3) selecting alternative catalogue components within a particular class of generic devices.

![Hierarchy of Domain-independent Functions](image)

**Figure 25** Organisation of design knowledge (Kota & Lee, 1993-1)

2.5.4 Discussion
TIPS can be applied to find a solution for a particular (small) design problem within production machines, such as improving a certain seal process. However, designing a complete machine will be impossible since it does not provide a product model of a production machine in which the found solutions can be incorporated.

Function taxonomies can be valuable to create a database with functions. This database should as well contain synonyms. The principle to express complex functions in terms of less complicated, elementary functions seems useful.

The catalogues used in catalog design can be used to create a database with solution principles. It would be convenient to search solutions based on the functions they realise since design is often the transition from function to form.

In general, the databases lack the presence of a product model that describes the artefact to be designed. Consequently, the found solutions or functions can not be incorporated into the product model as it exists at a certain point in time. To effectively use a database, the retrieved information should immediately be incorporated into the actual design.
2.6 Software tools to support conceptual design

This section focuses on computer tools that have been developed to support the conceptual design process. The aim is to determine the elements and approaches that can be used to develop computer support for the conceptual design of production machines. The functionalities of the tools are described very briefly because it would be too extensive to enumerate all the functionalities. First, tools that support the design process are given. Next, tools are discussed that more or less focus on one of the elements of the hypothesis: product model, design activities, and database. The section ends with a discussion about the reviewed tools.

2.6.1 Tools based on the design process

There have been several attempts in the 1970s to implement Methodical Design (section 2.2.1) into a computer-program. Because of the limitations of hardware and software, the programs were however only used to structure information. Kuttig (1993) describes some recent developments of software tools within German research.

PROSUS is a process-based design support system, which aims at improving the design process by using a process model as its core. This model captures the data resulting from design activities and supports the creation of these data throughout the process (Blessing, 1993, 1994 & 1996). The process model provides a framework for the activities of the designer. The basic building block of PROSUS is the Design Matrix that represents the design process as a structured set of issues and activities. Figure 26 shows a simplified version. Other elements of PROSUS are the product model, comments, the project file, procedure matrices, and the strategy matrix.
Göker & Birkhofer (1995) present a design system, Conceptual Design Assistant (CoDA), which uses both a theoretical and empirical approach. They combine the method and structure from the design methodology of Pahl & Beitz (1993) with experience-based problem solving and case-based reasoning (Figure 27). The case-based reasoning engine simulates the designer's experience. Methodical design is used to support the designer in understanding and analysing the designs in situations in which he does not have any experience. Besides that, methodical design serves as a grammar to structure the case-base for retrieval, adaptation, repair, and evaluation.
Hundal (1991) describes a computer program to perform systematic design. The program uses databases that contain solutions, which correspond to functions used in the function structure development. The function structure can be simplified by eliminating functions or connections (Hundal, 1991-2). Solutions are described at the level of abstraction of solution principles. The program presents solutions for a given function according to their input(s), output(s) and descriptions. The output of the program consists of alternative solution structures, which are evaluated in order to present the optimum concept. Figure 28 gives an example of solutions in the database. Hundal & Langholtz (1992, 1992-1) present screen-shots of the program.
Basic Function: Convert

<table>
<thead>
<tr>
<th>Physical Function</th>
<th>Input(s)</th>
<th>Output(s)</th>
<th>Solutions(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convert</td>
<td>Energy-Electrical</td>
<td>Energy-Mechanical</td>
<td>Electro Magnet E-Motor</td>
</tr>
<tr>
<td></td>
<td>-Mechanical</td>
<td>-Mechanical</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-Force</td>
<td>-Torque</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-Translating</td>
<td>-Rotating</td>
<td></td>
</tr>
<tr>
<td>Convert</td>
<td>Energy-Mechanical</td>
<td>Energy-Mechanical</td>
<td>Rack &amp; Pinion Linkage Brake</td>
</tr>
<tr>
<td></td>
<td>-Mechanical</td>
<td>-Mechanical</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-Force</td>
<td>-Torque</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-Translating</td>
<td>-Rotating</td>
<td></td>
</tr>
</tbody>
</table>

Figure 28 Excerpts from the solutions database (source: Hundal, 1991)

Suh & Sekimoto (1990) describe an implementation of Axiomatic Design (section 2.2.2). This Thinking Design Machine is based on a mathematical description of the mapping between functional requirements and design parameters and on the two axioms.

Commercially, very little tools are available to support the first stages of the design process. Two programs are mentioned here: TechOptimizer™ and Machine Lab™. The basis of these programs is the TIPS design technique (section 2.5.1).

TechOptimizer™ is used to define the design problem and consists of three modules:
- Function Analysis, to set up a functional model of the system,
- Trimming Technique, to simplify the model by eliminating harmful or less valuable features in the system and
- Feature Transfer, which analyses competing systems to find useful components that can be transferred to the design.

Machine Lab™ consists of following three modules:
- Invention Machine Principles™, to solve engineering contradictions,
- Invention Machine Effects™, to find solutions (effects) for a certain function and
- Invention Machine Prediction™, which can be used to solve problems that involve interactions between products.

2.6.2 Tools based on product models
A Designers Workbench (Mortensen 1993, Hansen 1997) is a design system able to support the designer in all design phases from specifying goals to detailing components. The chromosome model (section 2.3.2) is the core of the system. A critical point in the establishment is the filling in of knowledge. The
chromosome product model is designed to maintain the knowledge used in an actual design project, so the contents of this model and earlier models may be used as a knowledge base. Jensen (1997) describes the implementation as well as an initial evaluation of the Designer Workbench. He concludes that the implementation does not fit well to practical design. Further he states that the Theory of Domains and the Theory of Technical Systems are strong in a retrospective mapping of the design process, but may not explain and intent to support the synthesis activity.

Another computer-system which implements the Chromosome-model is described by Malmqvist (1995, 1995-2). The core of the system is a product model that consists of the chromosome model and a related function-means tree, see Figure 29. Malmqvist (1994) models the functions by using bondgraphs. Further, port specifications specify the in- and outputs of functions. The focus lies mainly on energy-transforming dynamic systems (Malmqvist, 1993 & 1994-2). Malmqvist & Schachinger (1997) describe the addition of design specifications (requirements).

![Design Process Management System](image)

**Figure 29** System framework (Source: Malmqvist, 1995)

Umeda et al. (1996) describes a computer implementation of FBS-modelling (section 2.3.4). The system can search for appropriate behaviours to a required function, check consistencies in the objective parts, and propose modifications for inconsistencies.

The metamodel as defined by Kiriyama et al. (section 2.3.5) is implemented in a prototype which uses following external design object modellers: a qualitative physics reasoning system, a solid modeller, a 2D draw modeller, a Function- Behaviour-State modeller and a beam modeller.
2.6.3 Tools based on design activities

The Unilever Research Laboratory at Vlaardingen, the Netherlands developed a computer-tool Modessa (Kersten, 1996). The underlying theory is the morphological chart (section 2.4.2), supplemented with databases with solutions and functions and weighting tables to select between solutions. The company used the program for several years. Vloemans, Roozenburg & Kersten (1997) describe an evaluation of the program.

Yu & MacCallum (1997) have developed two interrelated product structures, a Product Breakdown Structure (PBS) and Product Family Classification Trees (PFCTs), which are depicted in Figure 30. The PBS represents a conceptual product model in a hierarchy of a parts breakdown. Links in the structure are viewed as "a part of" and each node stands for a module, subsystem or elements. Each node in the PBS belongs to its own PFCT, in which a group of elements has highly commonality of functions, characteristics, or manufacturing process. Nodes in the PFCT have relevant parts breakdown at different levels of detail. The Product Breakdown Structure and the Product Family Classification Trees are the basis for the program ConfigMan, which is a "virtual configuration workbench" (Yu, 1996).

![Figure 30: The framework of product structures as defined by Yu (1996)](image)

Kota & Lee (1993-2) present an application of their proposed configuration design method (section 2.4.3) for the design of hydraulic circuits.
Potter et al. (1997) describe a system for the automatic configuration of hydraulic systems. The program uses machine learning techniques based on Artificial Neural Networks (ANNs) to acquisitive knowledge from past designs. The followed procedure is:

- prompt user to provide a set of requirements,
- select most suitable circuit template based on these requirements,
- recognition of some set of basic functions which the requirements indicate and ought to be achieved by candidate solutions,
- select components to achieve each of these basic functions and determine their positions within the chosen template and
- incorporate the chosen components within the template.

Expert systems are custom-written computer programs that are "expert" in some narrow problem area, and embody (to a certain extent) a true human expert's knowledge, experience, and problem-solving strategies. Expert systems have been used in many problem areas, such as medicine, chemistry, geology, meteorology, computer systems, etc. Expert systems can generally be used in problem areas that do not require common sense to be solved, that are well understood, and where human expertise is scarce. A requirement is that the input-data for the expert system can be described objectively. Limitations of expert systems are that they cannot easily adapt to new and unusual situations. In other words, they are not creative. Further, expert systems do not learn by experience, are not good at representing spatial knowledge, have no common sense, and are expensive and time-consuming to develop.

2.6.4 Tools based on databases
Goldhahn et al. (1990, 1991 & 1994) describe a computer-tool Vat-I (German: Verarbeitungstechnische Informationssystem) which has the aim to store and retrieve solution-principles for machine processes. Figure 31 depicts the underlying product model.

![Diagram of process components](image)

**Figure 31** Basic components within a process (Kurfürst & Majschak, 1993)
Both processed materials and functions are categorised. The main categories for the processed materials are: high viscous, low viscous, gas, rubble, single thread, and flat. The functions are categorised into: separate, join, mould, buffer, dose, transport, and arrange.

Data management systems concern all the tools that are dealing with large amounts of various data. Examples of these tools are Product Data Management (PDM) systems. PDM systems integrate isolated and task specific computer-aided (CA) systems. They offer a flexible framework to control and manage engineering data, documents and technical processes in order to enable effective co-operation and co-ordination of teams, data exchange and concurrent engineering (Abramovici & Gerhard, 1997). Commercially many PDM-systems are available. Figure 32 shows the concept of PDM-systems.

![Diagram of PDM concept](image)

**Figure 32** Fundamental data separation concept of PDM systems (Abramovici & Gerhard, 1997)

Current PDM technologies cover following functional capabilities:
- document management;
- product structure and configuration management;
- part family management and part classification;
- workflow and process management;
- project and program management.
2.6.5 Discussion
The majority of tools is only available as prototype. They were mainly developed to evaluate the underlying models and did not aim at a practical (commercial) implementation. Consequently, the user-interfaces are often not user-friendly. The tools are very generic because they were developed to support the design of a variety of products. Therefore, they can only limited be used to design production machines since they can not describe specific characteristics of production machines, e.g. the motions within the machine. The contents of the databases can be used to create a database for the design of production machines.

Commercially, only a limited number of tools is available that supports the conceptual design stage. TechOptimizer and Machine Lab can be applied to solve a particular problem within a production machine, but they can not be used to design a complete machine. The PDM systems mainly manage the documents created during design, but they do not support the creation of these documents.

In general, no tool supports the design of production machines during the conceptual design stage in a satisfactory manner.

2.7 Conclusions
The elements of the hypothesis (product model, design activities, and database) can be found in many approaches. Most of them mention only one or two elements. Others mention three elements, but they do that not explicitly. The models have often such a generality that they can be applied to design a wide range of products. However, this deteriorates the capability to model more specific characteristics of a product. The used terminology is often very general and academic and does not correspond to the terminology that designers use. Nevertheless, the models, methods, and tools have useful elements that can be used to develop design support for production machines.

The principle to describe a product by using multiple views connects highly to the designers' way of thinking. Views describing the specification, functions and solutions can be suitable to model production machines. Besides these views, views that show specific properties of a production machine should be developed.

The several design activities can be used to determine the type of required design activities for the design of production machines. These activities should connect to the product model and database.
Function taxonomies can be valuable to create a database with functions. This database should as well contain synonyms. The principle to express complex functions in terms of less complicated, elementary functions seems useful. The catalogues used in catalog design can be used to create a database with solution principles. It would be convenient to search solutions based on the functions they realise since design is often the transition from function to form. It must be possible to easily incorporate the found solutions or functions into the product model.
3

Supporting Conceptual Design

3.1 Introduction

Previous chapters described the initial problem statement and the state of the art. The problem statement showed that the conceptual design stage of production machines is hardly supported. The state of the art concluded that no solution is available that supports the design of production machines in a satisfactory manner, but that there a various models, methods and tools that contain suitable principles.

The hypothesis introduced in Chapter 1 is elaborated. The elements of the hypothesis are used to describe the research problem, requirements for design support, and the pursued design support in detail.

3.2 Development of a model of the design process

3.2.1 Conceptual design

During the conceptual design stage, the designer creates a description or model of the product to be designed. This product model contains all documented information about a design as it exists at a certain point in time. It can contain pictures, technical drawings, calculations, sketches, sideways notes, rejections, working methods, and so on. The designer creates this model by performing design activities. Examples are the addition, removal, and modification of solution concepts, performing an analysis, or calculating certain properties. Both the analysis and calculation result in information that can be added to the product model. Design activities can have different levels of complexity. Basic design activities can be used to create design activities that are more
complicated. For example, a calculation consists of an activity that performs the actual calculation and an activity that adds the results to the product model.

Figure 33 shows the product model, design activities, and the data that the design activities add to the product model.

![Figure 33 Example design activities on the product model](image)

The design process starts with a blank product model and during design, the designer incrementally creates this product model by performing activities. Figure 34 shows how a product model as it exists at time $i$ ($PM_i$) is modified by a design activity ($DA_i$), resulting in an evolved product model at time $i+1$ ($PM_{i+1}$). The chain of design activities can be seen as the design process.

![Figure 34 Evolving product model and executed design activities](image)
3.2.2 Reuse of information
This section elaborates on reuse of information. A distinction between data, information and knowledge is made, followed by a model of the reuse-process.

**Data, information, and knowledge**
Reuse can relate to data, information, and knowledge. Since these terms are often misused, an explanation is given.

Data is expressed by numbers, characters, images, or another type of recording. Examples are "6002479", "translate" and "図". Data on its own has no meaning, but when it is related to a meaning or a context, it becomes information. For example, 31/10/1997 is data, but if it represents a date of birth, then it is information. Information is the meaning conveyed by data. The meaning of data is often referred to as semantics and consequently, information is defined as data and its semantics. A *database* is a collection of logically related data (and a description of this data), designed to meet the information needs of an organisation (Connolly et al., 1999).

Knowledge is a collection of facts and inference-rules about these facts. Knowledge differs from information in that new knowledge can be created from existing knowledge by using logical inference. For example, knowledge about a family might contain the information that John is David's son and Tom is John's son and the rule that the son of someone's son is their grandson. From this, the new fact can be inferred that Tom is David's grandson. A knowledge base is a collection of logically related knowledge represented by using some knowledge representation language.

This thesis focuses on data and information since an exploration of knowledge would go beyond the scope of this research.

**Reuse of information**
Three elements are necessary to enable reuse of information: a database and storage-, and retrieval-processes (Figure 35). The arrows indicate the data that is stored or retrieved.

![Diagram](image)

**Figure 35** A database and the retrieval and storage of information

45
In the context of this thesis, the database contains background information that can be reused in new design cases. It not only holds the data but also the semantics given to this data. It is a repository of information, which is defined once and used by different users and in different design cases. The stored information must have such a degree of generality that it can be applied in many different designs. The storage process puts data in the database. The data must be unique to prevent redundancy in the database and the correctness of data must be checked before it is actually stored. The retrieval process gathers information from the database. An important part is the selection of the proper and desired information. To be able to select this information, the desired type of information and search-criteria must be given. The storage process changes the database, but not as frequently as design activities change the product model.

3.2.3 Model of the design process
Designers can create the product model by using information that is retrieved from the database. To enable this, the retrieved information must be incorporated into the product model. The product model of an existing design can contain information that can be reused in product models of new designs. Since not all information is general enough to be reused, it must be determined which information is suitable and how it can be extracted. It must be emphasised that the same database can be used in different design cases, so that information from one design can be reused in another design. Next figure combines conceptual design and reuse of information by concatenating Figure 33 and Figure 35. The arrows indicate the data that the activities generate or use.

![Diagram](image)

Figure 36 Conceptual design and the reuse of information
The activities in Figure 36 can be grouped, resulting in a general model of the (conceptual) design process as shown in Figure 37. The arrows represent data flows between the product model, design activities, and database. The design activities can modify the product model, database, or both.

![Model of the Design Process](image)

The model of the design process applies to the conceptual design of a wide range of products. Each type of product has a specific product model, design activities, and database. For example, a product model of a production machine will be different from a product model of a chair. Consequently, both products will have other design activities and databases. To use the design process for a certain application domain, a specific product model, design activities, and database must be developed.

### 3.3 Detailed research description

#### 3.3.1 Problem description

The initial problem statement showed that the conceptual design stage is hardly supported. The state of the art concluded that design support mainly focuses on one or two of the three elements of the model of the design process. For example, a database and corresponding retrieval procedures are defined, but a connection with a product model is lacking. Further, it stated that the models, design activities, and database have a number of deficiencies.

**Product model**

Designers generate and use various types of models during the design process. Examples are technical drawings, calculations, bill of materials, sketches, functional specifications, decisions, and so on. These different models describe the same design and therefore, they must be consistent with each other. For example, when a technical drawing contains a certain solution, this solution must somehow be reflected in the bill of materials. It is time consuming and difficult to manually maintain the consistency between the different models and consequently, the models are often not consistent.
The product models are often very abstract and academic. Further, they are very generic. Consequently, they can model a wide range of products, but they are not capable to describe specific properties of a particular type of product.

The models that designers use do often not have a predefined format. For example, a model showing the functions of a machine represents the functions at one time in a table, but at another moment, it represents them in a flowchart. Besides the different representations, the model contains different information in time. For example, the flowchart contains information about the sequence of functions, which is lacking in the table. The absence of predefined formats can lead to misunderstanding when designers communicate about a design and it decreases the potential reuse of information.

**Design activities**
Design activities can be very monotonous and time-consuming. Examples are manual calculations, drawing standard construction-elements like bolts, and browsing the company’s documentation to find a specific drawing. Due to this, mistakes are easily made, which can result in errors in the design.

Design activities often have no predefined format. Therefore, designers have to figure out the activities and consequently, activities are performed differently in time.

**Database**
The archive of a company contains models of designs that were generated in the past. Past designs contain geometric and non-geometric information, but often, only the geometric information is archived. Consequently, a considerable amount of non-geometric information contained in past designs is lost.

Past designs are rarely reused completely and often, only a part of a design is desired. For example, a designer only wants to reuse a conveyor belt and not a complete machine.

Retrieved information often has a format that can not directly be incorporated into a product model. An example is that the format of a solution in the database differs from the format in the product model. To be able to use the information, it has to be converted to a format that is compatible with the product model.
3.3.2 Requirements for support of conceptual design

The pursued design support should contain a product model, design activities, and a database. These elements must be integrated in a design support system. The problems mentioned in previous section must be solved, considering the following requirements.

Product model
The product model must give the designer a comprehensive understanding of a production machine. Therefore, the product model must connect to the models and terminology that the designers use. Since it presumably consists of several sub-models, the consistency between the models must be maintained. The product model must represent specific properties of production machines and it must contain geometric and non-geometric information. Since conceptual design deals with the transition from function to shape, it must as well reflect these aspects. Further, it must be able to deal with uncertain and incomplete information. It must have a predefined format, without deteriorating the capability to describe a range of production machines.

Design activities
The design activities must cover a considerable amount of actions that the designers performs during conceptual design. They must be described in a form that a design support system can support or automate. It must be defined what an activity realises and how it is performed. In a design support system, it must be transparent what the result of an activity will be. Further, the execution of automated or supported activities must be flexible so that they do not hamper the designer's way of working.

Database
The database must contain both geometric and non-geometric information about designs. This information must be relevant and applicable to the design of production machines. It must be enabled that only a part of a past design can be retrieved. The retrieved information must be incorporated in the product model without (manual) conversions. Queries must be defined that support the designer in searching information. It must be possible to oversee and select between the items found. The database should provide functionalities to add, update, or remove information.

3.3.3 Pursued support of conceptual design

The model of the design process serves as a basis to describe the pursued design support. Figure 38 shows the current and the supported design process. The current design process shows the current available product model, design activities, and database. The supported design process consists of two layers: a definition and an instance layer. The definition layer defines the format of the
product model, the procedures to perform design activities, and a data schema. The instance layer uses these definitions to instantiate an actual product model, design activities, and database. The current design process is used to develop the definition layer.

![Diagram of design process layers](image)

**Figure 38** Pursued support of the conceptual design process

The **meta product model** defines what data is documented in the product model, the relationships among this data, the constraints on the data, and the semantic information about the data. The meta product model can be seen as a model of the product model. The designer can use the meta product model to create and instantiate an actual product model. The **product model** contains all documented data about a design as it exists at a certain point in time. It must be emphasised that the developer of the support system creates the meta product model and that the designer uses this meta model to instantiate an actual product model.

A **design activity procedure** prescribes how an activity is performed, which input it requires and how it affects the product model, database, or both. A **design activity** is an executed (or instantiated) design activity procedure that results in an extension or modification of the product model, database, or both. A design support system can support or automate the execution of design activities.
A data schema describes what data is stored in the database, the relationships among this data, the constraints on the data, and the semantic information about data. A database is a collection of logically related data (and a description of this data) about designs. It contains background information about past designs that can be used in product models of new designs. The data schema defines the logical relation between the data. It must be emphasised that the database contains the actual data and that the data schema only defines the structure.

To develop design support for the conceptual stage, a meta product model, procedures of design activities, and a data schema for production machines have to be developed. This will be elaborated in chapters 4 to 6.

3.4 Conclusions

This chapter elaborated on how the conceptual design stage can be supported. A model of the design process has been developed that consists of three elements: a product model, design activities, and a database. The deficiencies of these elements were discussed and with additional conditions, they resulted in requirements for conceptual design support. The pursued support of conceptual design was introduced. This support consists of a definition and an instance layer. The definition layer consists of a meta product model, procedures of design activities and a data schema. By using the procedures of design activities, actual design activities are performed that create an instance of a product model, which is based on the meta product model. The data schema is used to create a database that can be consulted during design.

Next three chapters elaborate on the development of the meta product model, procedures of design activities, and data schema.
4

Meta Product Model

4.1 Introduction

4.1.1 Overview
This chapter elaborates on the product model and the meta product model (Figure 39). In current design process, the design is described by a product model that has several shortcomings, amongst others inconsistency between sub-models and non-uniform formats. To overcome these, a meta product model that supports the designer with the creation of a product model has to be developed.

Figure 39 Focus of this chapter
First, the product model used in current design process is analysed, which results in a direction for the meta product model. Next, a general meta product model is presented, as well as a method that describes how developers can create a meta product model for a specific product. The general meta product model is used to create a specific meta product model for production machines. The chapter ends with conclusions.

4.1.2 Analyses of current product models
This section elaborates on current product models. First, it is argued that a product model consists of multiple aspect models. Second, the non-uniform format of these models is elucidated. Third, the inconsistencies that can occur between these models are explained.

Multiple aspect models
It is very hard to define what the design at a certain point in time is. Designers use many different models to represent the design. At one moment they are creating a technical drawing, while at another moment they describe the functions the design must fulfil or they use some other kind of model. Each of these aspect models describes the product from a particular point of view and highlights a certain aspect of the design. Together, the aspect models form the product model (Figure 40). Designers interpret the aspect models within product model to create a mental image of the design.

![Figure 40](image_url)

**Figure 40** A design is described by a product model containing multiple aspect models

Since the product model in the supported design process must connect to the designer’s way of thinking, it should contain multiple aspect models. Currently used aspect models can serve as a basis to develop the meta product model. The aspect models must contain both geometric and non-geometric information and must use a terminology that connects to the terminology that designers use. Further, they must describe specific properties of production machines and must be able to contain uncertain and incomplete information.
Non-uniform format of the aspect models
The format of an aspect model consists of data, the structure of this data, and
the presentation of the data according to predefined specifications. Within
current product model, most aspect models do not have a predefined format and
consequently, aspect models are non-uniform in time. For example, at one
moment, an aspect model describing the functions of the product consists of
functions and their sequential relations, while at another moment it consists of
functions and their decomposition. Because of the absence of predefined
formats, three problems arise. First, the changing contents of an aspect model
leads to confusion and misunderstanding during the interpretation of the aspect
model. Second, aspect models are easily mixed up, e.g. a technical drawing
contains information about the functions of the design. Third, aspect models
with a predefined format are more and better documented as compared to
aspect models without a predefined format. However, since these
undocumented aspect models contain a reasonable amount of information about
the design, information is lost. These problems mainly occur with aspect
models describing non-geometric information, e.g. function structures, since
they do not have a predefined format. Aspect models that describe geometric
information often have a predefined format, which decreases the problems. For
example, technical drawings have numerous drawing conventions to obtain
uniform models.

Inconsistencies between aspect models
Since all aspect models describe the same design, it is possible that they contain
and present the same data. For example, one aspect model can present data in a
table while another presents the same data in a graph. It is obvious that, when
data changes in one of the aspect models, data in other aspect models must
change as well to maintain the consistency among the aspect models. The
process of propagating a change in an aspect model into changes in other aspect
models is called consistency management. Manual management of consistency
is very difficult, monotonous, and time-consuming. Therefore, it is often carried
out only partially and mistakes are easily made. Consequently, inconsistencies
in aspect models occur which means that they contain mutually conflicting
information.

4.1.3 Direction for improvements
This section elaborates on how the issues mentioned in previous section can be
solved.

Multiple aspect models
Each type of product requires its own specific aspect models. For a chair,
models of materials, costs, and aesthetics are important, whilst models of
motions, costs, and materials are important for a machine. Thus, different type
of products can have similar (materials and costs), but also very distinctive (motions and aesthetics) aspect models. Together, the aspect models must provide the designer with a comprehensive understanding of the design. It is however very hard to determine which aspect models are needed to achieve that. To give some indication of the required aspect models, three types are distinguished: problem definition models, concept models, and evaluation models. **Problem definition models** contain information about the problem to be solved and can contain a design problem, requirements, boundary conditions, required functions and so on. **Concept models** contain information about the concepts that solve the problem and can contain the geometry of solutions or a bill of materials. **Evaluation models** value to what extent the concept solves the problem. An example is the result of the calculation of stresses. Together, the models form the product model. Since the product model contains both the design problem and concepts and because it is not prescribed that one aspect model must be completed before another, problem and concept can be developed simultaneously.

![Figure 41 Different types of aspect models](image)

Several methods can be used to determine proper aspect models of a product. Interviewing and observing designers in combination with a literature study can give an indication of the required aspect models. Another manner is to determine the characteristics of the product. However, since it is impossible to completely forecast the required aspect models, the meta product model should be flexible so that new aspect models can be added. Section 4.2.2 describes a method to add aspect models to the meta product model.

**Uniform formats**
Aspect models in the improved product model must have predefined formats and must be consistent with each other. To describe predefined formats and consistency management, it is convenient to divide an aspect model into data and the presentation of data. Data carries information about the design, while presentations visualise this data. The presentation of the product model is very important because it informs the designer about the actual status of the design. If the presentation does not show the information in a right, convenient manner,
it can lead to errors and unneeded design steps. The presentations should give the designer an idea about the elements that describe the current design. In current product model, a rigid separation of data and presentations is often lacking and sometimes, the presentation is the data. This makes it very difficult to define a uniform format and to manage the consistency of the product model.

Aspect models in the improved product model must have a rigid distinction between data and presentations. A uniform format of the product model is obtained by using a predefined data schema and predefined presentations as defined by the meta product model. The data schema defines the data-types and their structure. During design, designers use this data schema to create the product model, which ensures a uniform data model. Predefined presentations define the visualisation of data and ensure a uniform presentation.

**Consistency management**

Since the product model consists of several aspect models, the meta product model must contain a mechanism that manages the consistency. Two mechanisms have been developed: direct and indirect consistency management.

Figure 42 shows *direct consistency management*. The meta product model consists of multiple aspect models and dependency relations. An aspect model consists of a data schema, presentations, and mapping rules. A *mapping rule* defines how a presentation relates to data and vice versa. It defines that if the data exist, the correlated presentations must exist as well and if data is lacking, the presentations may not exist. *Dependency relations* define how the data schemas in the different aspect models relate to each other. Direct consistency management is the process of propagating a change in an aspect model directly into changes in all other aspect models, based on dependency-relations. Since an aspect model directly relates to another aspect model, it is named direct consistency management.

![Diagram](image)

**Figure 42** Meta product model based on direct consistency management
The advantage of direct consistency management is that it is very transparent because the dependency relations exactly define how a change in one aspect model will affect other aspect models. A disadvantage is the number of required dependency relations. When the meta product model consists of \( n \) aspect models and when each aspect model relates to another, \( O(n^2) \) dependency relations are required. It is obvious that the number of dependency relations expands very rapidly when aspect models are added to the meta product model. Therefore, direct consistency management deteriorates the expandability of the meta product model. Another disadvantage is the redundancy of data, which means that the same data is contained in more than one aspect model. Although direct consistency management maintains the consistency of this data, it is inconvenient that data is stored more than once.

To overcome these problems, indirect consistency management has been developed. Instead that each aspect model contains and presents data, the data now exists only in one location and is presented in different views (Figure 43). A view only presents the data from a particular point of view and therefore, it only contains presentations and it does not contain data. A mapping rule defines how a presentation in a view corresponds to data and vice versa. Based on a mapping rule, a change in a view results in a change to the data. This change to data is propagated into changes to the presentations in the other views. The data schema and mapping rules define thus the consistency between the aspect models. Indirect consistency management is the process of propagating a change in a view indirectly into changes in all other views by changing the data in the data schema and presenting this change in all views. Since views indirectly correlate to each other, it is called indirect consistency management.

![Figure 43 Meta product model based on indirect consistency management](image)

An advantage of indirect consistency management is that each view only needs \( O(n) \) mapping rules. The expandability of the product model remains indifferent when the number of views increases: an addition of a view only needs an addition of \( O(n) \) mapping rules. Another advantage is the development of the data schema of the meta product model because it forces the developer of
design support to think more thoroughly about the similarities between aspect models. This results in a more compact product model and less redundancy of data. A disadvantage is that the propagation of changes is not very transparent because they are carried out through the data schema.

Based on the advantages, indirect consistency management is used throughout this thesis.

4.2 General meta product model

This section presents a formal notation to describe and a method to develop a meta product model.

4.2.1 Explanation of the formal notation

Meta product model
The meta product model consists of one data schema, multiple views and mapping rules. The data schema defines the structure of the data in the product model, views define how this data is visualised and the mapping rules relate the data schema to the views.

\[ mpm = \langle ds, v_1, ..., v_i, mr_1, ..., mr_i \rangle \]  [1]

Data schema
The data schema defines the structure of data in the product model. The data schema (ds) consists of a predefined number of collections and attributes. The brackets \( \langle \rangle \) indicate that the number of attributes and collections that describe a particular object is fixed within the definition of the meta product model.

\[ ds = \langle CO_1, ..., CO_n, a_1, ..., a_i \rangle \]  [2]

A collection (CO) contains a certain number of objects (o) from the same object-class. The braces \( \{ \} \) indicate that the number of objects within a collection can vary during the design process. For example, when the design process begins, a collection will contain no objects, but during the design process, objects are added to the collection. To distinct between a collection and an object, the name of a collection is written in capitals and the name of an object in small types.

\[ CO = \{ o_n \} \]  [3]
An object is a distinct entity (a person, place, thing, concept, or event) in the real world that is represented in the data schema. A number of attributes (a) describe the information content of an object-class.

\[ o = \langle ra_1, ..., ra_k, oa_1, ..., oa_t, da_1, ..., da_m \rangle \]  

An attribute describes a property of an object and can be numerical, textual, or graphical, but also a collection of objects or a reference to another object. Three types of attributes are distinguished: required, optional and derived attributes. Required attributes (ra) are necessary to describe the minimal information content of an object, e.g. a name or an identification key. Optional attributes (oa) expand the information content of an object, e.g. the colour and can have a default value, e.g. "green". Derived attributes (da) are determined from the required and optional attributes. Conditions (c) can be imposed on the attributes, e.g. \( c_1 : data > 12 \) or \( c_2 : data \in \{"begin","end"\} \).

\[ a = data \begin{cases} c_1 \\ \vdots \\ c_n \end{cases} \]  

Figure 44 shows the graphical representation of the data schema. A rounded rectangle visualises a collection of objects of a certain type. The line shows a relation between the object types. It indicates that one object of type A relates to exactly N objects of type B, and that one object of type B relates to zero or more objects of type A.

![Figure 44 Graphical representation of the data schema](image-url)

**Views**
A view (v) contains a number of presentations. A presentation (p) is a certain format to visualise data from the objects to which it is connected.

\[ v_i = \{ p_i \} \]  

\[ p_i = \text{(method of visualisation)} \]
Mapping rules
The objects within the product model must be presented to the designer. Therefore, a mapping rule (mr) relates a number of objects, specified by conditions, to a number of specific presentations. A mapping rule defines that if the objects exist, the presentations must exist as well and vice versa.

\[ mr_i: \left\langle o_1, \ldots, o_i \right\rangle^{dr} \Leftrightarrow \left\langle p_1, \ldots, p_n \right\rangle^{\text{view}} \]

\[
\begin{align*}
C_k \\
\end{align*}
\]

4.2.2 Method to develop the meta product model
Figure 45 shows a method to develop the meta product model. The method starts with defining an aspect model that consists of presentations, a data schema, and mapping rules. Definitions of objects and their information content describe the data schema.

If no meta product model is already available, the data schema becomes the data schema of the meta product model. Otherwise, the data schema of the new aspect model must be compared to the data schema of the meta product model (step 2). This is done by comparing the objects in the aspect model with the objects in the meta product model. This results in a number of object mappings, which define the relation between objects in the new aspect model and objects in the meta product model. Conditions specify the particular sets of objects. Expression [9] gives a general object mapping. If necessary, objects can be added to the data schema of the meta product model or the information-content of objects can be expanded by adding attributes.
\[ \langle o_1, \ldots, o_n \rangle^{\text{ds}} \Leftrightarrow \langle o_1, \ldots, o_l \rangle^{\text{model}} \]
\[ \left\{ \begin{array}{l}
\vdots \\
c_k
\end{array} \right. \] [9]

Since a view cannot contain data but only presentations, the mapping rules within a model are transformed into mapping rules between presentations in the view and the objects in the data schema of the meta product model (step 3).

\[ \langle o_m, \ldots, o_n \rangle^{\text{ds}} \Leftrightarrow \langle o_i, \ldots, o_l \rangle^{\text{model}} \]
\[ \langle o_i, \ldots, o_l \rangle^{\text{model}} \Leftrightarrow \langle p_m, \ldots, p_n \rangle^{\text{view}} \] [10]

By using this method, a theoretically infinite number of aspect models can be added to the meta product model, which makes the model very flexible to extensions.

### 4.3 Meta product model for production machines

The general meta product model has been used to develop a meta product model for production machines. This specific model is explained in this section.

#### 4.3.1 Production machines

Production machines produce similar products with the same quality at high production speeds and volumes. Examples are the production of light bulbs (approximately 1 bulb per second), packing of cookies, and the production of cans (20 cans per second).

A production machine can often be divided into several units that process the incoming materials to the required shape. Each of the units can be designated a specific machine function. Specific processes within the units, e.g. a filling or melting process, are in this thesis considered as a black box. Between the units, the materials are transported. A control unit ensures the correct quality and course of the production process.

Another characteristic of production machines is that there are many moving parts within the machine. Electronic control or a rigid mechanical coupling tune these motions to each other.
Figure 46 shows a machine for the production of nails that works as follows: a supply catch transports a metal thread. Next, the thread is fixed by a catch, punched, and finally cut by a knife. This example will be used throughout this chapter to explain the meta product model for production machines.

**Figure 46 Machine for the production of nails**

### 4.3.2 Aspect models

From the characteristics as described in previous section, it can be seen that a function- and solution model is desired. Further, an aspect model has to be defined that describes the materials processed by the machine. Similar models were found in the state of the art (Chapter 2). To describe the movements and transitions of parts within the machine, an aspect model that shows these transitions must be defined. It must also contain information about the relations between the transitions within the machine. Finally, an aspect model presenting how the solutions are positioned in space must be defined. Figure 47 depicts the developed aspect models. Next sections explain the different aspect models.

**Figure 47 Product model of a production machine**
4.3.2.1 Product state model

The product state model presents the states that the product to be produced has during the production process. This is presented in the aspect model by a product state structure. The elements in the product state structure are the states of the product and the relations between these product states. A product state describes the state of the product that the production machine produces, at a specific point in the production process. A name, textual description, and a picture can describe a product state. However, for reuse purposes, a formal definition of a product state is required. This is obtained by introducing properties of the product state and their values. A product state property describes a certain characteristic of the product. Examples are the position, orientation, and temperature. The product state property is specified by a value, e.g. the temperature is 20 degrees. The values can be textual ("high", "low") or numerical. The relations between states are time-relations and indicate the sequence of states in time. Figure 48 shows the product states for the nail producing machine. For clarity, the properties and their values are omitted.

![Figure 48 Product state model of the nail producing machine](image)

4.3.2.2 Function model

The function model presents the function structure of the production machine. A function structure consists of the functions and the relations between the functions. A function describes what must be achieved and is presented by a verb. The subject of a function is described by the product state. A function can be a transformation, but also the prevention of a spontaneous transformation of a product state. There are two types of relations between functions. A time-relation between functions indicates the sequence in time in which the functions occur. A requires-relation indicates that a function requires another function. In the example, the "deform" function requires the "fix" function. Figure 49 shows the function structure for the nail producing machine.
4.3.2.3 Solution model

The solution model presents the solution structure of the machine. The solution structure shows the solutions within the machine and the relations between solutions. A solution represents a part of the machine. It can presented as a CAD-drawing, 3D-model, or sketch. Besides geometric information, it can contain information about process conditions, advantages, disadvantages, estimated costs, reliability and so on. The difference between a function and a solution is that a function describes what has to be achieved while a solution describes how this is to be achieved. There are two types of relations between solutions. A time-relation between solutions indicates the sequence in time in which the solutions process the product. A requires-relation indicates that a solution requires an other solution for its proper functioning. In the example, the "Punch" requires a "Catch" to fix the thread. Figure 50 shows the solution model for the nail producing machine.
4.3.2.4 Solution layout model

The solution layout model shows the size and the spatial placement of the solutions. The purpose of the solution layout model is to show the spatial relations between solutions. Figure 51 depicts the solution layout model for the nail producing machine.

![Solution layout model of the nail producing machine](image)

**Figure 51** Solution layout model of the nail producing machine

Figure 52 shows the definition of a solution in the solution layout model.

![Definition of a solution in the solution layout model](image)

**Figure 52** Definition of a solution in the solution layout model
4.3.2.5 State transition model

The working of a production machine consists of actions or movements, which are realised by a transition of a solution. Some of these transitions must occur sequentially in time, some must take place simultaneously, and others may not occur simultaneously because of collision between parts. For example, the punch in the nail producing machine may not move down while the knife is closed.

To be able to design this complex transition pattern, a state transition diagram is used. A state transition diagram describes the state of the machine at discrete points in time, the transitions of the solutions, and the relations between these transitions. Figure 53 shows an example of a state transition diagram for a single solution. A solution within the diagram consists of properties, states, and transitions. A property describes a specific form or position that a solution can have. Examples are "up", "down", "open", "closed", and so on. A state describes which property the solution has at a certain point in time. It must be emphasised that a state and a product state are not the same objects in the aspect models. A transition indicates that a solution proceeds from one state to another. A transition correlates to a rest when the property does not change during the transition. A functional transition is a transition that realises a particular function. In Figure 53, the transition "down-up" correlates to the function "supply". At the end of the transitions, the solution returns to its begin state, which completes the cycle.

![State transition diagram of the supply catch](image)

**Figure 53** State transition diagram of the supply catch

A state transition diagram of a complete machine contains the state transition diagrams of the solutions within the machine and relations between these solutions (Figure 54). Following relations can exist: sequential-demands, concurrent-demands, and collision-indicators.
Figure 54 Complete state transition model of the nail producing machine

**Sequential-demands**
A sequential-demand indicates that a transition of one solution must occur after a transition of another solution. Figure 55 shows a part of a state transition diagram and a sequential-demand to indicate that transitions AB and CD must occur sequential in time. A sequential-demand can be incorrect, optimal, or correct. The demand is incorrect when it "points back" in time. The demand is optimal when CD connects directly to AB and it is correct, but not optimal, when time is "spoiled" between AB and CD.

Figure 55 Correctness of a sequential-demand
The correctness of a sequential-demand can be determined by comparing the states B and C of the involved transitions:

\[
\begin{align*}
B > C & \quad \Rightarrow \text{incorrect} \\
B = C & \quad \Rightarrow \text{optimal} \\
B < C & \quad \Rightarrow \text{correct}
\end{align*}
\]  

[11]

Concurrent-demands
A concurrent demand indicates that two transitions of different solutions must take place simultaneously. This is often the case when two solutions co-operate while performing a function. For example, the catch must be closed when the thread is punched, otherwise, the product will not maintain its position. Figure 56 shows a part of a state transition diagram and the concurrent-demand to indicate that transitions AB and CD must occur concurrent in time.

*Figure 56 Correctness of a concurrent-demand*

A concurrent-demand can be correct or incorrect. This depends on whether the destination transition is a rest or motion, as explained in Figure 57.
Figure 57 The correctness of a concurrent-demand depends on the destination transition

If destination transition CD is a motion (situations C and D), CD must take exactly as long as AB:

\[ A = C \land B = D \quad \Rightarrow \text{correct} \]  

[12]

If destination solution-transition CD is a rest (situations A and B), there are three possibilities, as depicted in Figure 58.

Figure 58 Possibilities when the transition CD is a rest

To determine which situation exists, following equations can be used:

\[ (A < B) \land (C < D) \quad \Rightarrow \text{situation 1} \]

\[ (A > B) \land (C > D) \quad \Rightarrow \text{situation 2} \]  

[13]

\[ (A < B) \land (C > D) \quad \Rightarrow \text{situation 3} \]
Each situation has another equation to determine its correctness. In situation 1, the concurrent-demand is correct when AB is completely within CD. In similar manner, the correctness in the other situations can be determined:

\[(C \leq A) \lor (B \leq D) \Rightarrow situation\ 1 = correct\]  \[14\]

\[(C \leq A) \land (B \leq D) \Rightarrow situation\ 2 = correct\]  \[15\]

\[(B \leq D) \lor (A \geq C) \Rightarrow situation\ 3 = correct\]  \[16\]

**Collision-indicator**

A collision indicator indicates that two solutions will collide when two transitions appear concurrent in time. Figure 59 shows a part of a state transition diagram and a collision-indicator to indicate that transitions AB and CD will collide when they occur at the same time. The left figure shows that AB and CD occur at the same time, which means that the solutions will collide. The right figure shows that AB and CD do not occur at the same time and that no collision occurs.

![Collision and No-Collision Diagrams](image)

**Figure 59** Correctness of a collision-indicator

The correctness of a collision-indicator can be determined by comparing the involved transitions. When the transitions do not overlap, the collision-indicator is correct. Because a transition can "go through a cycle border", three situations are possible (Figure 60).
To determine which situation exists, following equations can be used:

\[(A < B) \land (C < D) \Rightarrow \text{situation A}\]
\[(A > B) \land (C < D) \Rightarrow \text{situation B}\]
\[(A < B) \land (C > D) \Rightarrow \text{situation C}\]  \hspace{1cm} [17]

Each situation has another equation to determine whether transitions overlap. In situation A, the collision indicator is correct when AB is completely left or completely right of CD. Similar, the correctness can be determined for situations B and C:

\[(B \leq C) \lor (A \geq D) \Rightarrow \text{situation A correct}\]  \hspace{1cm} [18]
\[(B \leq C) \land (A \geq D) \Rightarrow \text{situation B correct, situation C correct}\]  \hspace{1cm} [19]

Through presentation of the correctness of the sequential demand, concurrent demand, and collision indicator in the state transition diagram, the designer is immediately informed if the diagram contains errors. This will support the designer in reducing the number of errors in a diagram and in designing it faster.

### 4.3.3 Data schemas of the aspect models

This section presents the data schemas of the aspect models. Table 3 gives a formal definition of the collections, objects, and attributes of the objects in the data schemas. For clarity, the restrictions on the attributes are not mentioned here, but given in Appendix A. Collections are described by capital letters and a collection "X" contains a number of objects "x". For example, the collection of product states (PS) contains a number of product states (ps).
### Product State Model

| Data schema | $psm$ | $= \langle PS, TRPSPS \rangle$ [20] |
| Product state | $ps$ | $= \langle \text{name}, \text{picture}, \text{PSP} \rangle$ [21] |
| Product state property | $p$ | $= \langle \text{name}, \text{picture}, \text{value} \rangle$ [22] |
| Time-relation between product states | $trp$ | $= \langle ps_{src}, ps_{dst} \rangle$ [23] |

### Function Model

| Data schema | $fum$ | $= \langle FU, TRFUFU, RR \rangle$ [24] |
| Function | $fu$ | $= \langle \text{name} \rangle$ [25] |
| Time-relation between functions | $trfu$ | $= \langle fu_{src}, fu_{dst} \rangle$ [26] |
| Requires-relation between functions | $rr$ | $= \langle fu_{src}, fu_{dst} \rangle$ [27] |

### Solution Model

| Data schema | $som$ | $= \langle SO, TRSOSO, RR \rangle$ [28] |
| Solution | $so$ | $= \langle \text{name}, \text{picture} \rangle$ [29] |
| Time-relation between solutions | $trso$ | $= \langle so_{src}, so_{dst} \rangle$ [30] |
| Requires-relation between solutions | $rr$ | $= \langle so_{src}, so_{dst} \rangle$ [31] |

### Solution Layout Model

| Data schema | $slm$ | $= \langle SO \rangle$ [32] |
| Solution | $so$ | $= \langle x_{org}, y_{org}, b_{org}, \text{length, height} \rangle$ [33] |

### State Transition Model

| Data schema | $stm$ | $= \langle SO, SD, CD, CI, cycle\_time \rangle$ [34] |
| Solution | $so$ | $= \langle \text{name}, SP_j, SS_j, ST_j \rangle$ [35] |
| Property | $sp$ | $= \langle \text{name} \rangle$ [36] |
| State | $ss$ | $= \langle sp, stepnumber \rangle$ [37] |
| Transition | $st$ | $= \langle ss_{src}, ss_{dst}, type \rangle$ [38] |
| Sequential-demand | $sd$ | $= \langle st_{org}, st_{dst}, correctness \rangle$ [39] |
| Concurrent-demand | $cd$ | $= \langle st_{org}, st_{dst}, correctness \rangle$ [40] |
| Collision-indicator | $ci$ | $= \langle st_{src}, st_{dst}, correctness \rangle$ [41] |

Table 3 Data schemas of the aspect models
Figure 44 gives a graphical representation of the data schemas.

![Graphical representation of data schemas](image)

Figure 61 Graphical representation of the data schemas

4.3.4 Data schema of the meta product model
The data schemas of the aspect models have to be integrated into one data schema of the meta product model. The meta product model must contain all the information from the separate aspect models as well as the information about the relations between these aspect models. The method to develop a meta product model (section 4.2.2) has been applied to develop the data schema of the meta product model. Table 4 shows the formal definition of the data schema of the meta product model. The restrictions on the attributes are omitted here, and mentioned in Appendix A.
<table>
<thead>
<tr>
<th><strong>Meta Product Model</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data schema</td>
</tr>
<tr>
<td>Function</td>
</tr>
<tr>
<td>Solution</td>
</tr>
<tr>
<td>Property of a solution</td>
</tr>
<tr>
<td>State of a solution</td>
</tr>
<tr>
<td>Transition of a solution</td>
</tr>
<tr>
<td>Product state</td>
</tr>
<tr>
<td>Property of a product state</td>
</tr>
<tr>
<td>Time relation between a function and a product state</td>
</tr>
<tr>
<td>Time relation between a solution and a product state</td>
</tr>
<tr>
<td>Functional relation</td>
</tr>
<tr>
<td>Relation between a functional relation and a transition of a solution</td>
</tr>
<tr>
<td>Requires relation</td>
</tr>
<tr>
<td>Collision indicator</td>
</tr>
</tbody>
</table>

Table 4 Data schema of the meta product model
Figure 62 gives a graphical representation of the data schema of the meta product model. The existence of some objects depends on the existence of other objects. For example, a time relation between a function and product state can only exist when both the function and product state exists. The arrows in Figure 62 indicate these dependencies between the objects.

4.3.5 Mapping rules
This section describes how the aspect models map on the meta product model. This is done by describing how objects in the aspect models map to objects in the meta product model. The mappings show that each object in an aspect model can be derived from information in the meta product model. In other words, it shows how information in the meta product model can be filtered and presented in a separate aspect model.

Figure 63 visualises a part of the meta product model of the nail producing machine that is obtained by integrating the information from the product state, function, and solution model. This example will be used to explain the different object mapping rules. For clarity, the conditions that specify the objects in the mapping rules are omitted, but they are given in Appendix A.
4.3.5.1 Mapping with the product state model

A product state in the meta product model maps directly to a product state in the product state model. A time relation between product states in the product state model can be derived in two different manners. It can be obtained via a function (the solid line in Figure 64) or via a solution (the dashed line). The figure shows that the time relation in the product state model is obtained by concatenating (1) a time relation between a product state and function, (2) a function, and (3) a time relation between a function and product state. In similar manner, the time relation between product states can be derived via a solution. Conditions specify the involved objects.
Expressions [56] to [58] formalise these object mappings.

\[ \langle ps_j \rangle^{pum} \Leftrightarrow \langle ps_i \rangle^{mpm} \]  [56]

\[ \langle trpsps_m \rangle^{pum} \Leftrightarrow \langle trfups_i, fu_j, trfups_k \rangle^{mpm} \]  [57]

\[ \langle trpsps_m \rangle^{pum} \Leftrightarrow \langle trsops_i, so_j, trsops_k \rangle^{mpm} \]  [58]

4.3.5.2 Mapping with the function model

A function in the function model maps directly to a function in the meta product model. Similar to the product state model, a time relation between functions in the function model maps to one product state and two time relations between a product state and function in the meta product model, as depicted in Figure 65. A requires relation in the product state model maps on a functional relation and a requires relation in the meta product model.

Figure 65 Mapping between the function model and the meta product model

Expressions [59] to [61] formalise the mappings between the function model and the meta product model.

\[ \langle fu_j \rangle^{fun} \Leftrightarrow \langle fu_i \rangle^{mpm} \]  [59]

\[ \langle trfufu_m \rangle^{fun} \Leftrightarrow \langle trfups_i, ps_j, trfups_k \rangle^{mpm} \]  [60]

\[ \langle rr_m \rangle^{fun} \Leftrightarrow \langle fr_i, rr_i \rangle^{mpm} \]  [61]
4.3.5.3 Mapping with the solution model

A solution in the solution model maps directly to a solution in the meta product model. Figure 66 depicts how a time relation (solid line) and a requires relation (dashed line) between two solutions can be obtained.

![Diagram showing mapping between solution model and meta product model](image)

**Figure 66** Mapping between the solution model and meta product model

Expressions [62] to [64] formalise these object mappings.

\[
\langle s_{o_j} \rangle^{som} \leftrightarrow \langle s_{o_j} \rangle^{mpm} \quad [62]
\]

\[
\langle trsos_{o_m} \rangle^{som} \leftrightarrow \langle trsops_{i}, ps_{j}, trsops_{k} \rangle^{mpm} \quad [63]
\]

\[
\langle rr_{r_m} \rangle^{som} \leftrightarrow \langle fr_{i}, rr_{r_j}, fu_{k}, fr_{r_i} \rangle^{mpm} \quad [64]
\]

4.3.5.4 Mapping with the solution layout model

A solution in the solution layout model corresponds to a solution in the data schema. This mapping is formalised by:

\[
\langle s_{o_j} \rangle^{slm} \leftrightarrow \langle s_{o_j} \rangle^{mpm} \quad [65]
\]

4.3.5.5 Mapping with the state transition model

A solution in the meta product model maps to a solution in the state transition model. A sequential demand maps on nine objects in the meta product model (Figure 67). By using a relation between a functional relation and transition (1), a functional relation (2) is found. This functional relation indicates the function
(3) that is realised by the origin transition. By following the time relation between a function and product state (4), a product state (5), and a time relation between a product state and function (6), a function (7) is reached. This function is used to determine the functional relation (8), the relation between a functional relation and transition (9), and finally, the destination transition. The origin and destination transitions completely define the sequential demand in the state transition model.

![Figure 67 Relation between the state transition model and meta product model](image)

In similar manner, it can be determined that a concurrent-demand in the state transition model maps to six objects in the meta product model. A collision indicator in the state transition model maps directly to a collision indicator in the meta product model.

Expressions [66] to [69] define the object mappings between the state transition model and the meta product model.

\[
\langle s_o_i \rangle^{stm} \Leftrightarrow \langle s_o_i \rangle^{npm} \quad [66]
\]

\[
\langle s_d_i \rangle^{stm} \Leftrightarrow \left\langle \begin{array}{c}
\text{rfrst}_{\text{org}}, \text{fr}_{\text{org}}, \text{fu}_{\text{org}}, \text{trfu}_{\text{org}} \\
\text{ps}_{i}, \text{trfu}_{\text{dist}}, \text{fu}_{\text{dist}}, \text{fr}_{\text{dist}}, \text{rfrst}_{\text{dist}}
\end{array} \right\rangle^{npm} \quad [67]
\]

80
\[ \langle cd_i \rangle^{sim} \iff \left\{ \frac{rfrst_{org}, fr_{org}, \ frr_i, \ fu_{dist}, fr_{dist}, rfrst_{dist}}{\ mpm} \right\} \]  

\[ \langle ci_j \rangle^{sim} \iff \langle ci_i \rangle^{mpm} \]  

4.4 Conclusions

A product model describes the design at a certain point in time. Product models that designers currently use contain many different aspect models, each highlighting a certain aspect of the design. Problems that arise are the inconsistency of data in these aspect models and the non-uniform format of the aspect models. To solve these problems, a meta product model has been developed.

The meta product model consists of one data schema, multiple views, and mapping rules. The data schema defines the structure of the data in the product model. A view visualises the data from a particular point of view and results in the visualisation of a certain aspect. The mapping rules define how the data relates to the visualisation in the views.

Since the meta product model is used to instantiate a product model during design, the product model will have a uniform data model. Since the mapping rules uniquely define how the views are derived from a (uniform) data model, the views are consistent.

A formal notation to define a general meta product model and a method to develop a meta product model for a specific type of products, are introduced. The formal notation and method are applied to develop a meta product model for production machines. This meta product model consists of five complementary models that describe specific characteristics of production machines. The product state model presents the states that the product to be produced has during the production process. The function model presents the functions that the design must realise and the sequence in which they must be realised. The solution model shows the sub-solutions within the machine and the sequence in which they transform the product. The solution layout model visualises the form and the spatial placement of solutions. Finally, the state transition model shows the transitions that the solutions perform to realise a function, and how these action transitions must be related to each other for a proper functioning of the machine.
5

Procedures of Design Activities

5.1 Introduction

5.1.1 Overview
This chapter elaborates on design activities (Figure 68). In current design process, the designer performs various design activities. Some are time-consuming, repeating, monotonous, and difficult, others are attractive and require much creativity. The objective of this chapter is to model activities in a way that a design support system can support or automate them.

Figure 68 Focus of this chapter
First, the design activities in current design process are analysed and directions for improvements are indicated. Next, procedures of general design activities are presented. These procedures are applied to develop procedures of specific design activities for the design of production machines. The chapter ends with conclusions.

5.1.2 Analyses of design activities
Designers perform many different design activities, varying from drawing a single line to the development of the operation-manual of a machine. They can be non-attractive, monotonous, and time-consuming. Examples are manual calculations, drawing solutions, and browsing the company’s documentation to find a specific drawing. Due to a lack of attention and time, mistakes are easily made, which results in errors in the designed product model. If a design support system can support or even take over activities, the number of errors made during design activities can be reduced. Besides, designers have more time left for other, more attractive activities.

Design activities often have no predefined procedure. Therefore, designers repeatedly have to figure out the activities and consequently, time is lost and activities are performed differently over time.

Design activities can have different levels of complexity. Basic design activities can be used to create design activities that are more complicated. For example, reuse is modelled by the less complicated design activities extract, store, retrieve, and incorporate. Further, a design activity can initiate another design activity. For example, an "add" activity is often followed by an "draw" activity.

5.1.3 Direction for improvements
In the supported design process, a design support system automates or supports design activities. This provides a uniform execution of activities, reduces the number of errors, and saves time. Kusiak, Szczerbicki & Vujosevic (1991) state that an intelligent system should not replace a human specialist, but effectively carry out specific tasks, that might form critical or timely aspects of a larger problem or project. In other words, the system should work with specialists to improve their output, rather than attempt to replace them. The followed approach follows this statement.

It is very hard to determine which set of activities the designer requires. During the development of the design support system, designers must be involved to verify the developed activities. To give the developer an indication of the possible activities, design activities are categorised in following groups: administration, presentation, synthesis, analysis, and evaluation (Figure 69).
Figure 69 Categories of design activities

Administration activities deal with the documentation of the product model. Examples are the activities add, remove, edit, and manage consistency. By using these activities, the designer can add objects to, remove objects from and edit objects in the product model. Consistency management maintains a proper mapping of the presentations in the different views and the data in the product model.

Presentation activities deal with presenting the product model to the designers. Examples are draw, undraw and redraw.

Synthesis activities generate information that can be added to the product model. Various synthesis activities can be recognised. Examples are the creation of an object from scratch and searching objects in a database. The result of a synthesis activity is often a set of objects that can be added to the product model.

Analysis activities examine a certain aspect of the product model. Various analysis activities can be distinguished. Examples are the simulation of dynamic behaviour and a check whether a function in the product model is realised by a solution.

Evaluation activities compare the result of an analysis with the problem definition and judges the outcome. An example is the comparison of the overall cost of a design with the costs specified in the problem definition. The distinction between analysis and evaluation activities is often very vague and therefore, they are frequently integrated.

The different activities have a strong correlation with each other. For example, when an object has been modified, this change should immediately be reflected in the presentation. The process that one activity initiates the execution of other activities is called chaining. Two activities are chained when one activity initiates the execution of another. If a design support system automates
chaining of activities, the developer of this system must be cautious that circular chains can occur. This is explained more in section 5.2.6.

Several methods can be used to determine a proper set of design activities. During the development of design support, designers should be observed and consulted to verify the developed design activities. Since it is impossible to completely forecast the required design activities, it must be possible to extend the number of activities in the design support system.

5.2 General design activities

This section elaborates on general design activities that support or automate the design of a product model as defined in Chapter 4. These design activities can be applied to develop specific design activities for the design of production machines.

5.2.1 Administration activities

Add

The product model can be created by adding objects. For example, a solution-object can be added to the product model of production machines. Objects are added to the product model by providing the set of required and optional attributes. The add-activity checks whether these attributes are conflicting with the restrictions imposed on them. If there are no conflicts, a new object is created with the provided attributes and added to a parent-object, which is often the product model. An add-activity has following form:

\[
\text{parentobject.add_object(\text{required attributes}, \text{[optional attributes]})} \quad [70]
\]

Remove

Objects are removed from the product model by removing them from a collection of objects that describes the product model. The removal of an object can initiate the removal of other objects when their existence depends on this object. The remove-activity must first remove the objects which existence depends on the object to be removed, before removing the actual object that had to be removed. For example, Figure 70 shows the removal of object B. Since the existence of link AB depends on the existence of B, it is removed before B is removed.

(1) Removing B  (2) Requires removing AB  (3) Before B is removed

Figure 70 Example of removing an object that initiates the removal of an other object

86
The remove-activity has following form:

```
parentobject.remove_object (object)
```

[71]

**Edit**

Objects in the product model are edited by assigning a new value to an attribute of the object. For example, a function-object is edited by assigning a new value for its name. If the new value conflicts with the restrictions imposed on the attribute, the modification is rejected and the object remains indifferent. The edit-activity has following form:

```
object.attribute = new_value
```

[72]

**Manage consistency**

The manage-activity maintains the consistency between the objects in the product model and the presentations in the views. It consists of two types of activities: project and reconstruct. The project-activity modifies the views when the data in the product model is modified. The reconstruct-activity modifies the data in the product model when a view is modified. These activities are based on the mapping rules that define the relation between the data in the product model and the presentations in the views.

The designer mainly designs in the views. Since the views only contain presentations, a modification of a view must be reconstructed into modifications of the data in the product model. Figure 71 shows an intended modification (1) to a view. This modification can not directly be applied to the view and therefore, it is reconstructed (2) to a modification in the data of the product model. This modification is then projected (3) to an actual modification to the view, resulting in the intended modification.

![Figure 71 Reconstruction of a change in a view](image-url)
The reconstruct-activity is more complicated than the project-activity since a modification in a view can often be reconstructed to distinct modifications to the data in the product model, which after projection result in the same intended modification to the view.

The execution of the project-activities depends on the type of modification to the product model (add, remove, or edit). The reconstruct-activities depend on the type of modification to the view (draw, undraw, or redraw). Figure 72 shows the different types of project- and reconstruct activities.

![Diagram](image)

**Figure 72 Different types of projections and reconstructions**

To explain the project- and reconstruct-activities, the example mapping of equation [73] will be used. This mapping defines that if instantiated objects of classes \(y\) and \(z\) exist in the product model and if they satisfy restrictions \(c_1\) and \(c_2\), presentation \(yz\) must exist in the view and vice versa.

\[
\langle yz \rangle_{\text{view}} \iff \langle y, z \rangle_{\text{pmp}} \quad \begin{cases} 
  c_1 \\
  c_2 
\end{cases} \tag{73}
\]

When an object is *added* to the product model, the project-activity checks for each mapping whether the addition of an object completes a combination of objects described by this mapping. If a combination is found, the related presentation is created in the proper view. For example, if an object of class \(y\) is added, it is checked whether a combination \((y, z)\) is formed that satisfies conditions \(c_1\) and \(c_2\). If this combination is found, presentation \(yz\) is created in the view.

When an object is *removed* from the product model, the project-activity checks for each mapping whether the removed object completed a combination of objects described by this mapping. If this combination is found, the related presentation is removed from the proper view. For example, if an object of
class y is removed, it is checked whether a combination (y,z) can be found that satisfies conditions c₁ and c₂. If this combination is found, the related presentation yz is removed from the view.

When an object in the product model is edited, the project-activity checks for each mapping whether the edited object belongs to a combination of objects described by this mapping. If this combination is found, the related presentation is edited according to the modification of the object. For example, if an object of class z is edited, it is checked whether a combination (y,z) exists that satisfies conditions c₁ and c₂. If this combination is found, the presentation yz is edited according to the modification of object z.

When a presentation is drawn in a view, the reconstruct-activity checks for each mapping which objects should exist in the product model to have a consistent mapping. If an object is lacking, it must be added to the product model to make the mapping consistent. When this addition is automatically performed, the default values of the required attributes of the object will be used to create the object. For example, after addition of presentation yz it is checked whether both y and z exist. If y does not exist, it must be added to the product model. The attributes of y are assigned the default values.

When a presentation is undrawn in a view, the reconstruct-activity checks for each mapping which objects are related to this presentation. If the set of objects described by the mapping exists, one or more objects within the set must be removed from the product model to make the mapping consistent. Thus, multiple possibilities arise to propagate the modification to the view. For example, after removing presentation yz, it is checked whether both y and z exist. If they both exist, the possibilities to make the mapping consistent are removing only y, removing only z, or removing both y and z. The selection between the possibilities depends on the type of mapping and the objects involved.

When a presentation is redrawn in a view, the reconstruct-activity checks for each mapping whether the set of related objects exists. If this set exists, one or more objects within the set must be edited. Again, multiple possibilities arise to propagate the modification. For example, after editing presentation yz it is checked whether both y and z exist. If they exist, the possibilities to reflect the changes are editing only y, editing only z, or editing both y and z. The selection between possibilities depends on the type of mapping and the objects involved.

5.2.2 Presentation activities
Three types of presentation activities are distinguished: draw, undraw and redraw. The draw-activity creates a presentation in a view. This presentation
depends on the values of the attributes of the objects to which it is related. The undraw-activity removes a presentation from a view. The redraw-activity updates an existing presentation in a view, which is needed when an attribute of the related objects changes. The activities have following forms:

\[
\begin{align*}
\text{presentation.draw_view} & \quad [74] \\
\text{presentation.undraw_view} & \quad [75] \\
\text{presentation.redraw_view} & \quad [76]
\end{align*}
\]

### 5.2.3 Synthesis activities

Synthesis activities depend heavily on the type of product to be designed. General synthesis activities are searching information in a database and adjusting an attribute of an object.

**Searching information in a database**

There are many different ways to search information in a database. The procedures can be divided into browsing and searching. Browsing is the process of manually looking through the database. Searching is the process of (automatically) looking through the database in a thorough way in order to find something that is related to certain input-data. Browsing mainly leads to results when the database is reasonably small and when it has a structure that simplifies browsing. Searching requires input and results in output. The input can be a certain keyword, but also an object or set of objects. By using one or more tables in the database (the database is explained in Chapter 6), the output is generated, which is often a set of similar items that match the input. A search-activity has following form:

\[
\langle \text{input} \rangle \rightarrow [\text{used tables}] \rightarrow \{\text{output} \}
\]    

[77]

An example of a search-activity is to find all people of entity-type "Persons" that are 45 years of age.

**Adjusting attributes of objects**

Adjusting attributes of objects means that a target attribute of an object is optimised by changing attributes from other objects that determine this target attribute. The difference between the adjust attribute- and the edit-activity is that the adjust-activity is often based on certain rules that add some "intelligence" to the procedure and that it uses edit-activities of other objects to adjust the target attribute. The adjust-activity has following form:

\[
\text{object.adjust_attribute} \quad [78]
\]

An example of the adjustment of an attribute is the optimisation of the strength and weight of a design by adjusting the strength and weight of the individual objects in the design.
5.2.4 Analysis activities
Analysis activities depend on the type of product to be designed. Examples of analysis activities are the simulation of dynamic behaviour and the calculation of the overall cost of a design. The activities have following form:

\[ \text{object.analyse\_attribute} \]  

[79]

5.2.5 Evaluation activities
Evaluation activities can relate to single or multiple objects. When it relates to a single object, a certain attribute of the object is compared with a given specification in the problem definition. This attribute is often obtained by an analysis activity. When evaluation relates to multiple objects, it compares these objects based on the problem specification. Well-known means to perform such an evaluation are weighting-tables. The type of evaluation activities depends on the type of product to be designed. The activities have following form:

\[ \text{object.evaluate\_attribute} \]
\[ \text{evaluate(set\_of\_objects)} \]  

[80] [81]

5.2.6 Design activities initiating other design activities
Design activities are chained when one design activity initiates another. Equations [82] to [91] show possible chains. Chain [91] indicates that the removal of one object initiates the removal of other objects. Examples are the removal of child-objects when a parent-object is removed.

- Add/Remove/Edit  \rightarrow Manage Consistency (project)  
- Draw/Undraw/Redraw  \rightarrow Manage Consistency (reconstruct)  
- Manage Consistency (reconstruct)  \rightarrow Add/Remove/Edit  
- Manage Consistency (project)  \rightarrow Draw/Undraw/Redraw  
- Add/Remove/Edit  \rightarrow Synchronise (project)  
- Add/Remove/Edit  \rightarrow Synchronise (project)  
- Add/Remove/Edit  \rightarrow Analyse  
- Analyse  \rightarrow Evaluate  
- Synthesise  \rightarrow Add/Remove/Edit  
- Remove  \rightarrow Remove

Attention should be paid to circular chains. For example, when chains [82], [85], [83], and [84] are executed in this order, it can happen that this chain infinitely continues. If circular chains occur, proper termination rules must be defined. An example of a termination rule is to determine whether a presentation was drawn manually by the designer (reconstruction is required and the chain continues) or whether it was automatically drawn (no reconstruction required and the chain terminates).
5.3 Design activities for production machines

This section gives an overview of design activities that can be used to create a product model for a production machine. These activities are supported or automated by the prototype design support system (Appendix B).

5.3.1 Administration activities

Add

Objects can only be added to the product model, the solution object, or the product state object. After addition, the consistency is managed, which results in the creation of the related presentations. The available add-activities are enumerated in Table 5.

<table>
<thead>
<tr>
<th>Product model</th>
<th>Solution</th>
<th>Product state</th>
</tr>
</thead>
<tbody>
<tr>
<td>ds.add_fu (name)</td>
<td>so.add_sp (name)</td>
<td>ps.add_psp (name)</td>
</tr>
<tr>
<td>ds.add_so (name)</td>
<td>so.add_st (ss_org, ss_dst)</td>
<td></td>
</tr>
<tr>
<td>ds.add_ps (name)</td>
<td>so.add_ss (sp, stepnumber)</td>
<td></td>
</tr>
<tr>
<td>ds.add_trfps (fu, ps, origin)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ds.add_trsops (so, ps, origin)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ds.add_fr (fu, so)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ds.add_rfret (fr, st)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ds.add_rr (fr, fu)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ds.add_ci (st_1, st_2)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5 Add-activities for production machines

Remove

Objects can only be removed from the product model, solution object or product state object. If the object to be removed contains child-objects, they are removed first. For example, if a solution-object has to be removed, the solution-property-objects it contains are removed first. After removal, the consistency is managed by undrawing the related presentations. The available remove-activities are enumerated in Table 6.

<table>
<thead>
<tr>
<th>Product model</th>
<th>Solution</th>
<th>Product state</th>
</tr>
</thead>
<tbody>
<tr>
<td>ds.remove_fu (fu)</td>
<td>so.remove_sp (sp)</td>
<td>ps.remove_psp (p)</td>
</tr>
<tr>
<td>ds.remove_so (so)</td>
<td>so.remove_st (st)</td>
<td></td>
</tr>
<tr>
<td>ds.remove_ps (ps)</td>
<td>so.remove_st (ss)</td>
<td></td>
</tr>
<tr>
<td>ds.remove_trfps (trfps)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ds.remove_trsops (trrops)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ds.remove_fr (fr)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ds.remove_rfret (rfret)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ds.remove_rr (rr)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ds.remove_ci (ci)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6 Remove-activities for production machines
Edit
Each object can be modified by altering one of the attributes of the object. However, before the proposed modification is propagated, it must be validated that the new value does not violate the restrictions imposed on the attributes. These validations can be rather simple, like checking whether the new value is an integer. In some cases, they are more complicated, which is the case when the cycle time of a state transition diagram and the stepnumber of a solution state are modified.

The cycle time of a state transition diagram can be unlimited high. However, the minimum value depends on the number of transitions that solutions within the product model contain. For example, the "Supply Catch" in Figure 73 contains four transitions. Since the minimum duration of a transition is one step, the minimal number of steps is 4. If a new value is assigned to the cycle time, it must be larger than the number of transition within each solution.

![Diagram](image)

**Figure 73** Example of a state transition diagram

The new value of the stepnumber of a state has two restrictions. First, the new value may not already be assigned to any other state. Second, the new value must be adjacent to current value. For example, in Figure 73, the steps without a state are 0, 1, 5, and 7. The only adjacent value for state C is 5, so this is the only possible new value for state C. Adjacency can however be more complicated. For example, when adjacent steps for state A must be determined, it can easily be seen that steps 0 and 1 are adjacent. However, due to the periodicity, step 7 is adjacent as well. In similar manner can be determined that state D has adjacent steps 0, 1, 5, and 7.

Management of consistency
The project- and most of the reconstruct-activities are rather straightforward and will not be elaborated here. However, some reconstruct-activities need more explanation and for that purpose, mapping [92] will be used.

\[
\langle \text{tr}psps_m \rangle_{p,m} \iff \langle \text{tr}ops_i, so_j, \text{tr}ops_k \rangle_{p,m}
\]

[92]
In the case that presentation \( \langle trpsps_m \rangle \) is drawn in the product state view, it must be reconstructed in such way that all objects \( \langle trsops_i, so_j, trsops_k \rangle \) exist in the product model. This can be achieved by adding all the objects. However, if objects \( \langle trsops_i, so_j \rangle \) already exist, only object \( \langle trsops_k \rangle \) has to be added to reconstruct the modification. Figure 74 shows situations A and B and their presentation in the view. After drawing the presentation in the view, situation A has possibilities A1 and A2 to reflect this modification, while situation B has possibilities B1, B2, and B3. It cannot be predicted which of the possibilities should be chosen. Therefore, it is chosen to add all the objects to the product model, which relates to situations A1 and B1 in the example.

![Figure 74 Possibilities while reconstructing](image_url)

When a presentation is undrawn, one or more related objects in the product model must be removed. The option to delete all the objects is too coarse since some of the objects contain much information. Consider for example mapping [92] and the case that \( \langle trpsps_m \rangle \) is undrawn. If all objects \( \langle trsops_i, so_j, trsops_k \rangle \) would be deleted to maintain consistency, much information is lost since \( \langle so_j \rangle \) has a large information content. Therefore, it is chosen to remove objects \( \langle trsops_i, trsops_k \rangle \) since their information content is much smaller. In general, the objects that have the smallest information content, which are often the objects that denote a relation between other objects, are deleted. In mapping rules [93] to [100], which were introduced in section 4.3.5, the objects that are removed from the product model when a presentation in a view is undrawn, are written bold. It must be emphasised that these mapping rules are those in which a presentation maps to more than one object.

\[
\langle trpsps_m \rangle^{pm} \leftrightarrow \langle trfps_i, fu_j, trfps_k \rangle^{pm} \quad [93]
\]

\[
\langle trpsps_m \rangle^{pm} \leftrightarrow \langle trsops_i, so_j, trsops_k \rangle^{pm} \quad [94]
\]
\[
\begin{align*}
\langle \text{trfufu}_m \rangle^{\text{sum}} & \leftrightarrow \langle \text{trfups}_i, ps_j, \text{trfups}_k \rangle^{\text{mpm}} \quad [95] \\
\langle \text{rr}_m \rangle^{\text{sum}} & \leftrightarrow \langle \text{fr}_i, \text{rr}_j \rangle^{\text{mpm}} \quad [96] \\
\langle \text{trsozo}_m \rangle^{\text{sum}} & \leftrightarrow \langle \text{trsozo}_i, ps_j, \text{trsozo}_k \rangle^{\text{mpm}} \quad [97] \\
\langle \text{rr}_m \rangle^{\text{sum}} & \leftrightarrow \langle \text{fr}_i, \text{rr}_j, \text{fu}_k, \text{fr}_l \rangle^{\text{mpm}} \quad [98] \\
\langle \text{sd}_i \rangle^{\text{sum}} & \leftrightarrow \langle \text{rfrst}_\text{org}, \text{fr}_\text{org}, \text{fu}_\text{org}, \text{trfups}_\text{org}, ps_i, \text{trfups}_\text{dst}, \text{fu}_\text{dst}, \text{fr}_\text{dst}, \text{rfrst}_\text{dst} \rangle^{\text{mpm}} \quad [99] \\
\langle \text{cd}_i \rangle^{\text{sum}} & \leftrightarrow \langle \text{rfrst}_\text{org}, \text{fr}_\text{org}, \text{fr}_i, \text{fu}_\text{dst}, \text{fr}_\text{dst}, \text{rfrst}_\text{dst} \rangle^{\text{mpm}} \quad [100]
\end{align*}
\]

When a presentation is redrawn, this modification can be reconstructed in a modification to the object with the largest information-content. For example, if the \text{cd}_i in [100] changes, it can be reconstructed to a modification of \text{fu}_\text{dst}, which has the largest information-content. Since these processes are rather straightforward, it will not be elaborated more.

### 5.3.2 Presentation activities

Each of the presentations in Table 7 has its own draw, undraw, and redraw procedure. If the designer executes one of these activities, the proper reconstruct-activity is initiated.

<table>
<thead>
<tr>
<th>Product state view</th>
<th>ps</th>
<th>trpups</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution view</td>
<td>so</td>
<td>trsozo</td>
<td>rr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Solution layout view</td>
<td>so</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function view</td>
<td>fu</td>
<td>trfufu</td>
<td>rr</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State transition view</td>
<td>so</td>
<td>sd</td>
<td>cd</td>
<td>ci</td>
<td>sp</td>
<td>ss</td>
</tr>
</tbody>
</table>

Table 7 Overview of views and their presentations
5.3.3 Synthesis activities

Several synthesis activities have been developed. This set of activities is not complete and must continuously be extended.

Searching information in the database

Each of the tables in the database can be browsed. Besides, various search-activities have been developed, from which the most useful will be elaborated.

Expression [101] shows an activity that searches solutions that can realise a certain function. In expression [102], this search-activity is extended by adding information about the product-state, which is used to filter the found solutions in such way, that only solutions that have been connected to this product state are returned.

\[
\langle fu \rangle \rightarrow [\text{functional realisation}] \rightarrow \{so\} \quad [101]
\]

\[
\langle fu, ps \rangle \rightarrow [\text{functional realisation}, \text{time relation solution}] \rightarrow \{so\} \quad [102]
\]

Expression [103] describes an activity that searches required functions for a functional relation. This can be helpful to extent the design with functions that a solution requires to realise its function.

\[
\langle fr \rangle \rightarrow [\text{requires relation}] \rightarrow \{fu\} \quad [103]
\]

The last example is an activity that searches for a state transition diagram when a solution is given. This supports the designer in creating a state transition diagram for a single solution.

\[
\langle so \rangle \rightarrow [\text{solution property}, \text{solution state}, \text{solution transition}] \rightarrow \{sp\}, \{ss\}, \{st\} \quad [104]
\]

**Draw morphological overview**

The search-activities can be extended by creating specific overviews of the found items. One of these overviews is a morphological chart. The draw-morphological-chart activity has multiple input-functions and repeatedly executes the search-activity described in equation [101]. The result is an overview of the input-functions and the alternative solutions that can realise these functions (Figure 75).
<table>
<thead>
<tr>
<th>Functions</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move</td>
<td>Conveyor Belt</td>
</tr>
<tr>
<td>Fix</td>
<td>Gripper</td>
</tr>
<tr>
<td>Fill</td>
<td>Nozzle</td>
</tr>
</tbody>
</table>

Figure 75 Example of a morphological overview

Eliminate insignificant steps in the state transition diagram
An insignificant step in a state transition diagram is a step that contains no states and only transitions. Figure 76 shows a state transition diagram with two insignificant steps. These steps are not required for the functioning for the machine, but they deteriorate the readability of the diagram and they unnecessarily increase the cycle time of the diagram.

Figure 76 Example of useless steps

The eliminate insignificant step activity determines for each step whether it is insignificant by examining whether it contains states. If it is insignificant, the step is removed.

Auto modify
In some cases, the product model can automatically be modified, which is explained in Figure 77. Situation A contains a time relation between a function and a product state (trfups) and a functional relation (fr). Based on the definition of the meta product model, it can be derived that there are two possibilities to modify the product model. Relation trsops(psl,sol) can be added (situation B) or trfups can be removed (situation C). The selection between these possibilities can only be made when the activity “responsible
for" or preceding situation A is known. In case that \textit{trfps} or \textit{fr} was added, it is logical to add \textit{trSops} as well. If \textit{trSops} was removed, it is logical to delete \textit{trfps}. When preceding activity is not known, the designer should give the decisive answer.

![Figure 77 Example of auto-modifying the product model](image)

There is one pitfall with auto-modify, which is explained in Figure 78. In situation D it is tempting to add \textit{fr(fu1,so1)}, which results in situation E. However, it could be the case that the designer intended to obtain situation F. Because of this pitfall, the automatic addition and removal of a functional-relation is not automated in the design support system.

![Figure 78 Example of a pitfall with auto-modify](image)
The rules behind auto-modify can be described in similar manner as object-mappings and are presented by expressions [105] and [106]. These rules can be used to derive the appropriate actions when objects are added or removed.

\[
\langle \text{trfups}_i, \text{fr}_j \rangle_{\text{m}} \Rightarrow \langle \text{trsops}_k \rangle_{\text{m}}
\]

\[
\begin{align*}
\text{fr}_j \cdot \text{fu} &= \text{trfups}_i \cdot \text{fu} \\
\text{trsops}_k \cdot \text{so} &= \text{fr}_j \cdot \text{so} \\
\text{trsops}_k \cdot \text{ps} &= \text{trfups}_i \cdot \text{ps}
\end{align*}
\]  

[105]

\[
\langle \text{trsops}_i, \text{fr}_j \rangle_{\text{m}} \Rightarrow \langle \text{trfups}_k \rangle_{\text{m}}
\]

\[
\begin{align*}
\text{fr}_j \cdot \text{so} &= \text{trsops}_i \cdot \text{so} \\
\text{trfups}_k \cdot \text{fu} &= \text{fr}_j \cdot \text{fu} \\
\text{trfups}_k \cdot \text{ps} &= \text{trsops}_i \cdot \text{ps}
\end{align*}
\]  

[106]

5.3.4 Analysis activities

Several analysis-activities have been developed. This set of activities is not complete and must continuously be extended.

Simulate

The simulate-activity combines the state transition- and solution layout view. During simulation, the presentations in the solution layout view change according to the transitions in the state transition diagram. The result is that the movements of the solutions are animated and a visual check of the co-operation of solutions can be made.

Check functional realisations

It can occur that the product model contains functions that are not realised by a solution. The check functional relations activity checks whether each function is realised by a solution and presents the result to the designer.

Check state transition diagram

Within the state transition diagram, several demands have been distinguished. Sequential demands show the sequence of transitions, concurrent demands show that transitions must occur simultaneously in time, and collision-indicators show that two solutions will collide if transitions occur simultaneously in time. These demands can be correct or incorrect. For example, if the transitions of a sequential demand do not occur sequentially in time, the relation is incorrect, otherwise, it is correct. The check state transition diagram activity determines for all demands whether they are correct or not. These verifications are not complex, but they have to be made continuously for a large number of demands. The result is presented to the designer by changing the presentations of the demands.
Calculation of the functional efficiency of the machine

Within the state transition diagram, transitions realising a function determine the transformations of the product state. The remainder of the transitions is required to complete the cyclic transition of solutions. Figure 79 shows that only two out of five steps contain a functional transition. In three steps, the machine is moving without affecting the product, which is a loss of production time.

![State transition diagram to explain the functional efficiency of the machine](image)

Figure 79 State transition diagram to explain the functional efficiency of the machine

The functional machine efficiency is the percentage of steps in the cycle time in which the machine realises a function that transforms the product. Since this efficiency is calculated by using the steps and not the actual time, it is called functional machine step efficiency. This efficiency should be as high as possible and can be calculated by:

$$\text{functional machine step efficiency} = \frac{\text{number of functional steps}}{\text{cycle time in steps}} \times 100\% \quad [107]$$

The calculation activity determines for each step whether it contains a functional transition and calculates the functional machine step efficiency by using [107].

5.3.5 Evaluation activities

Current design activities for production machines do not contain evaluation activities. The main reason is that the meta product model defines the design problem only in terms of functional- and product-state descriptions. Consequently, the design problem is not described by numerical values, which makes a supported evaluation very difficult. The comparison of concepts or
solutions by using weighting tables is not investigated further. However, it can be incorporated in followed approach.

5.4 Conclusions

Designers perform a range of design activities during design. Some of these activities are very attractive and require a lot of creativity. Others are not interesting, monotonous, and time-consuming. Due to a lack of attention and time, mistakes are easily made during the execution of these routine activities, resulting in unneeded errors in the final design. This is enforced by the absence of strict procedures of the design activities. To solve these problems, procedures of general design activities have been developed. A procedure of a design activity prescribes how an activity is performed, which input it requires, and what the output will be. This output is often a modification to the product model. The procedures of design activities are categorised into administration, presentation, synthesis, analysis, and evaluation activities.

The categories and general procedures are applied to develop specific design activities for production machines. The administration and presentation activities support the designer in documenting and visualising the product model. Besides, they maintain the consistency of the data and the presentations in the product model. The synthesis activities support the designer in finding information in the database and in optimising the product model. The analysis activities check the correctness of demands in the state transition diagram and simulate the working of the machine in the solution layout model. Presenting the correctness of demands to the designer assists him to solve potential errors in the state transition diagram of the machine. The simulation supports him to obtain a more realistic image of these transitions. The evaluation activities are as yet executed by the designers.

The automated and supported activities assist the designer to design a product model faster and more easily because changes to the model are automatically maintained and presented. The activities assist in finding information in the database, which could stimulate the creativity of the designer. Design activities can detect potential errors and can optimise the product model, which results in better designs.
6 Data schema of Background Information

6.1 Introduction

6.1.1 Overview
This chapter elaborates on databases that contain background information about designs that a designer can use while creating the product model (Figure 80). It must be emphasised that this thesis focuses on databases that store information about designs and not on databases that store information about the design process.

![Diagram of database schema](image)

*Figure 80* Focus of this chapter
In current situation, available databases that contain information about designs have several shortcomings. To overcome these, the developer of a design support system has to develop a data schema that supports the designer during the creation of an improved database.

First, available databases in current design process are analysed and directions for improvements are indicated. Next, a general data schema is developed based on common database techniques. This general data schema is applied to develop a specific data schema that supports the design of production machines. The chapter ends with conclusions.

6.1.2 Analysis of current databases
In current design process, different sources with background information about designs are available that can be used to find information that can be added to the product model. First, there is the company's documentation of past designs, which mainly contains technical drawings and manuals about designs. Second, experienced designers have information and knowledge they can share with inexperienced designers. Third, a large amount of information is available in common literature, the internet, and so on. However, these sources of information have several shortcomings and limitations.

Limited storage of non-geometric information
Designers use many different types of information to create the product model. Examples are information about solution principles, the functions a solution can realise, calculation procedures, past designs, and information about how to proceed the design process. In current design process, only information that is very concrete, e.g. technical drawings, CAD models, or manuals, is stored for reuse. Other, more abstract information, like a functional description, is stored less frequently. Consequently, a considerable amount of information contained in these models is lost.

Non-uniform formats of information
Information is often stored in different formats. For example, a solution is at one time archived as a 3D-model and at another time stored as a bill of materials. This occurs even more with non-geometric information, like function-structures or a problem specification, since this information does not have a predefined format. Different formats deteriorate the storage and retrieval, and consequently, the potential reuse of information.

Incorporation of retrieved information
The information that is retrieved from the database must be used in the product model. However, two issues deteriorate an easy incorporation of information. First, retrieved information has often a format that can not directly be
incorporated into the product model. For example, the database describes a solution by a picture containing its name and properties, while the product model requires the name and properties in a textual format. Consequently, the designer has to convert the retrieved information. In general, information retrieved from databases requires a conversion to be incorporated in the product model. This is enforced by a possible physical separation of database and the environment in which the designer creates the product model. Second, information retrieved from the database often contains complete designs. However, mostly only a part of the design is desired and the designer has to manually extract the specific information. For example, a designer wants to reuse a conveyor belt from a design of a production machine and he manually has to extract the information about the conveyor belt from the design of the production machine.

Sharing information by designers
Psychological factors and the engagement of designers to a company influence the exchange of information between designers. When designers leave a company, their knowledge leaves as well. In the past, employees tend to work within the same company for a long period, but nowadays, employees switch companies more often and almost on a regular basis. Therefore, uniform storage of information in permanent and accessible media is required.

Diversity of information
Much information is available in common literature, catalogues, manuals, and so on. However, the information is often very divers and can not directly be used in a product model. Consequently, designers have to convert the information to a format that can be used in the product model. Further, it is difficult and time-consuming to find the desired information.

6.1.3 Direction for improvements
In the supported design process, the improved database must contain both geometric and non-geometric information in a uniform format, and the retrieved information must easily be incorporated into the product model.

The uniform format is ensured by a data schema that defines the structure for the database. The data schema is specified during the development of the design support system and does not change frequently. However, the database that is instantiated from the data schema, may change frequently.

The ability to incorporate retrieved information into the product model is achieved by relating the data schema of the database to the meta product model (Figure 81). In that way, the database relates to the product model and shall contain similar entities as the product model.
The meta product model contains predefined formats for objects describing geometric and non-geometric information. Since the data schema is derived from the meta product model, it will contain these uniform formats and types of information as well.

6.2 General data schema

This section elaborates on data schemas. The ANSI-SPARC architecture is presented, which shows the several levels of a database. Further, the conceptual and logical data schema are explained as well as transactions on these data schemas.

6.2.1 The three-level ANSI-SPARC Architecture

The ANSI-SPARC architecture (ANSI, 1975) is developed to obtain a standard terminology and general architecture for database systems. Although the architecture was developed in 1975, its concepts are actual and commonly used. The objective of the three-level architecture is to separate each user's view on the database from the way it is physically represented.
The external-level is the users' view of the database. It describes that part of the database that is relevant to each user. An external-view includes only those entities, attributes, and relationships in the 'real world' that the user is interested in.

The conceptual-level describes the community-view of the database. This level describes what data is stored in the database and the relationship among data. It is introduced as a layer between the several user views and the internal schema. A conceptual and a logical data schema describes the conceptual level. The conceptual data schema describes the structure of the database at a high level of abstraction and is independent of the type of data model that the target database system uses (for example a relation or an object-oriented data model). The conceptual data schema is transformed into the logical data schema to obtain a more precise and concrete structure that depends on the type of data model of the target database-system.

The internal-level is the physical representation of the database in the computer. This level describes how the data is stored in the database. Below the internal-level there is a physical-level that is managed by the operation system.

This chapter focuses on the conceptual level of a database.

6.2.2 Conceptual data schema
There are many methods to describe a conceptual data schema. In this thesis, entity-relationship-modelling (ERM) as defined by Chen (1976) is used because of its simplicity and because it is commonly accepted and used. The entity-relationship model consists of entity and relationship types and of instantiated
entities and relationships. It can graphically be represented by an entity-relationship diagram.

**Entity- and relationship types**
The basic concept of the entity relationship model is an entity-type, which represents a set of objects in the real world with the same properties. Relationship-types are meaningful associations among entity-types, often representing relations between the objects in the real world. A data schema normally contains many different entity- and relationship-types.

An *entity-type* is a distinct type of object (e.g. a person, place or thing, concept or event) in the real world that is to be represented in a data schema and that has an independent existence. Each entity-type is identified by a name and a list of attributes. An attribute is a property of an entity-type. Each attribute is associated with a set of values called a domain. The domain defines the potential values that an attribute may hold. A key is a data item that uniquely identifies individual occurrences of an entity type.

A *relationship-type* is a set of associations between two (or more) participating entity-types. Each relationship-type is given a name that describes its function. Similar to entity-types, a relationship-type can have attributes which describe a property of the relationship-type.

The *degree of a relationship-type* is the number of participating entity-types in a relationship-type. The most common degree of a relationship-type is two and is called binary.

**Entities- and relationships**
An *entity* or entity-instance is a uniquely identifiable occurrence of an entity-type. Although an entity-type has a distinct set of attributes, each entity has its own values for each attribute. The attributes of an entity hold the values that describe each entity. The values held by attributes represent the main part of the data stored in the database.

A *relationship* or relationship-instance is a uniquely identifiable occurrence of a relationship-type. A relationship is an association of entities where the association includes one entity from each participating entity-type.

The *cardinality-ratio* describes the number of possible relationships for each participating entity. The most common cardinality-ratios for binary relationship-types are one-to-one (1:1), one-to-many (1:M), and many-to-many (M:N).
**Entity-relationship diagram**

An ER-diagram shows the entity- and relationship-types in a graphical manner. Each entity-type is shown as a rectangle, labelled with the name of the entity-type (Figure 83). Attributes are shown as ellipses, labelled with the name of the attribute, and are connected to the entity-type to which they belong. A name of a key attribute is underlined. Each relationship-type is shown as a diamond, labelled with the name of the relationship-type. The participating entity-types in each relationship-type are connected by lines, which are labelled with 1, M or N, as determined by the cardinality-ratio or the relationship-type.

![Figure 83 Example of an entity-relationship-diagram](image)

**6.2.3 Logical data schema**

The conceptual data schema may contain data structures that are not easily modelled by data schema of the target database-system. Therefore, the structures must be transformed into a data schema that is more easily handled by the database-system. Connolly, Begg & Strachan (1999) give a method to perform this transformation. The result of a transformation is the logical data schema. Since the prototype design support system uses Microsoft Access®, the logical data schema is described by a relational schema.

The relational schema is described by entity-types that consist of attributes. A candidate key is an attribute or a minimal set of attributes that uniquely identifies individual occurrences of an entity-type. It must hold values that are unique for every occurrence of an entity-type. Since entity-types can have more than one candidate key, a primary key must be selected from the candidate keys. A foreign key is an attribute or a set of attributes within an entity-type that matches the primary key of some another entity-type. Foreign-keys are defined in the domain of their primary keys. A foreign key usually represents a relationship between entity-types. The common convention for representing a relation schema is to give the name of the entity, followed by the attribute names in parentheses. The primary key is underlined, and a foreign key is represented in *italic*. Figure 84 shows the logical data schema for the example.

109
6.2.4 Transactions

Transactions represent "real world" events such as registering a property for rent, the creation of an appointment to view a property by a prospective renter, the addition of a new employee, and the registration of a new client. These transactions have to be applied to the database to ensure that data held by the database remains current with the "real world" situation and to support the information needs of the users. A transaction is a single action, which may realise a series of actions as a whole that accesses or changes the content of the database. It must have following, so-called ACID, properties (Haerder & Reuter, 1983):

- Atomicity The "all or nothing" property. A transaction is an indivisible unit that is either performed in its entirety or it is not performed at all.
- Consistency A transaction must transform the database from one consistent state into another consistent state.
- Isolation Transactions execute independently of one another. In other words, the partial effect of incomplete transactions should not be visible to other transactions.
- Durability The effects of a successfully completed (committed) transaction are permanently recorded in the database and must not be lost because of a subsequent failure.

There are three main types of transactions: retrieval transactions, update transactions and mixed transactions. Retrieval transactions are used to retrieve data for display on the screen or in the production of a report. Update transactions are used to insert new records, delete old records, or modify existing records in the database. Mixed transactions involve both the retrieval and updating of data. For example, the operation to search for and display the details of a property and then update the value of the monthly rent.

A formal language to describe transactions is SQL (Structured Query Language). SQL was originally devised in the 1970s as a language for relation databases and has since become the de facto industry standard language for relational systems (ISO, 1992). With SQL, the user can manipulate data in the database by using commands like SELECT for retrieving information, INSERT
for creating information, **UPDATE** for altering information, and **DELETE** for removing data.

The developer of the design support system must develop both the data schema and the set of permitted transactions. The designer can use the data schema to initiate a database and use the defined transactions to perform operations on this database. It is obvious that the data schema and transactions depend on the type of product. Next section will elaborate on the data schema and transactions for production machines.

### 6.3 Data schema for production machines

#### 6.3.1 Conceptual data schema

The data schema of the meta product model for production machines served as basis to develop the conceptual data schema. The object-classes in the data schema of the meta product model were used to define entity-types in the conceptual data schema. For each object, it has to be determined which general information it contains. Consider for example Figure 85. The meta product model describes a product state by a number of product state-properties. In that way, a product state "A" can be described by e.g. the position, shape, temperature, and so on. In the meta product model, these properties belong to one specific product state. However, they can as well be used to describe a product state "B" in a different design. Therefore, a property of a product state in the database can be used to describe more than one product state in the database, as shown in Figure 85. The generality of objects has to be determined for each object and for each relationship in the data schema of the meta product model.

![Figure 85 Derivation of the conceptual data schema from the meta product model](image)

Additional entity-types and relationships to categorise information about entity-instances can be added to the data schema. The following categorisations will be mentioned: categorising function and categorising solutions.
Functions can be categorised by putting them into categories with similar functions, for example functions that alter similar properties of a product state. In this manner, a tree-structure of functions is obtained, as shown in Figure 86.

![Category of functions and Function tree](image)

**Figure 86** Example of the tree-structure of functions

Solutions can have several levels of detail, as shown in Figure 87. During the design process, the detail level of solutions will become more concrete. The final goal is to obtain standard components. The categorisation of solutions has the advantage that designers can start very abstract and that the database can support them in concretising the design. Furthermore, they can look up in the tree to abstract the solution. Next, they can potentially enlarge the solution-space, by looking down in the tree. For example, if a product model contains an electric actuator, the designer can look up in the tree, which results in the actuator. If the designer looks down in the tree, he finds the electric, but also the pneumatic actuator.

![Detail levels of solutions](image)

**Figure 87** Example of detail-levels of solutions
The entity-types retrieved from the meta product model supplemented with the additional entity-types from the categories define the conceptual data schema for production machines. Figure 88 shows the ER-diagram. The additional entity-types and relationships are hatched.
Figure 88 Conceptual data schema for production machines
6.3.2 Logical data schema

Figure 89 shows an example of the transformation from the conceptual- to the logical data schema. The transformation mainly concerned the removal of M:N and recursive relationships.

Figure 89 Transformation from the conceptual into the logical data schema

Figure 90 to Figure 93 show the additional entity-types and relationships to remove recursive and M:N relationships.

Figure 90 Additional entity-types for the product state
Figure 91 Additional entity-types for the solution

Figure 92 Additional entity-types for the category of functions

Figure 93 Additional entity-type for solution states- and transitions
Figure 94 shows the entity-types and their attributes in the logical data schema for production machines.

<table>
<thead>
<tr>
<th>Entity-Type</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>(FunctionID, FunctionCategoryID, name)</td>
</tr>
<tr>
<td>FunctionalRelation</td>
<td>(FunctionID, SolutionID)</td>
</tr>
<tr>
<td>FunctionCategory</td>
<td>(FunctionCategoryID, name)</td>
</tr>
<tr>
<td>FunctionCategoryRelation</td>
<td>(FunctionCategoryID, Parent, FunctionCategoryID, Child)</td>
</tr>
<tr>
<td>ProductState</td>
<td>(ProductStateID, name, picture)</td>
</tr>
<tr>
<td>ProductStateDescription</td>
<td>(ProductStateID, ProductStatePropertyID, ProductStatePropertyValueID)</td>
</tr>
<tr>
<td>ProductStateProperty</td>
<td>(ProductStatePropertyID, name, picture)</td>
</tr>
<tr>
<td>ProductStatePropertyValue</td>
<td>(ProductStatePropertyValueID, ProductStatePropertyID, name)</td>
</tr>
<tr>
<td>RelationFunctionalRelationTransition</td>
<td>(FunctionID, SolutionTransitionID)</td>
</tr>
<tr>
<td>RelationStateTransition</td>
<td>(SolutionTransitionID, SolutionStateID)</td>
</tr>
<tr>
<td>RequiresRelation</td>
<td>(FunctionID, FunctionalRelationID)</td>
</tr>
<tr>
<td>Solution</td>
<td>(SolutionID, name, picture, length, height, beta)</td>
</tr>
<tr>
<td>SolutionDescription</td>
<td>(SolutionID, SolutionPropertyID)</td>
</tr>
<tr>
<td>SolutionProperty</td>
<td>(SolutionPropertyID, name, picture)</td>
</tr>
<tr>
<td>SolutionRelation</td>
<td>(SolutionID, Parent, SolutionID, Child)</td>
</tr>
<tr>
<td>SolutionState</td>
<td>(SolutionStateID, SolutionID, SolutionPropertyID)</td>
</tr>
<tr>
<td>SolutionTransition</td>
<td>(SolutionTransitionID, SolutionID, RelationStateTransitionOrgID,</td>
</tr>
<tr>
<td></td>
<td>RelationStateTransitionDstID)</td>
</tr>
<tr>
<td>TimeRelationFunctionProductState</td>
<td>(FunctionID, ProductStateID, origin)</td>
</tr>
<tr>
<td>TimeRelationSolutionState</td>
<td>(SolutionID, ProductStateID, origin)</td>
</tr>
</tbody>
</table>

**Figure 94** Logical data schema for the design of production machines

### 6.3.3 Transactions

The developed data schema makes numerous transactions possible. Especially transactions dealing with the retrieval of information are valuable to support the design of production machines. Two examples of a select-transaction will be given.

The first example selects solutions that can realise a function. In Figure 95, input (A) is a reference to a function. This reference is used to find all functional relations in which this function occurs. These functional relations
point to solutions (B) that can realise this input function (A). Figure 95 also shows the SQL-statement to perform this transaction.

```
SELECT *
FROM Solution
WHERE Solution.SolutionID IN
  (SELECT SolutionID
   FROM FunctionalRelation
   WHERE FunctionID = A);
```

**Figure 95** Procedure to search solutions

The second example finds a set of functions (C) that are necessary to let a solution realise the input-function (A). Figure 95 shows this transaction with a dashed line. The transaction selects functional relations, which are used to select a number of requires-relations. These requires-relations are used to find the set of required functions.

### 6.4 Conclusions

The database contains background information that can be used in a product model. The shortcomings of current databases are the limited storage of non-geometric information, the non-uniform formats of the information, and the difficult incorporation of retrieved information. To solve these problems, a general data schema is introduced. This data schema is derived from the data schema of the meta product model and is described by common database techniques. Since the meta product model describes both geometric an non-geometric information in a uniform format, the data schema does this as well. Further, the entities in the database relate highly to the objects in the product model, which enables an easy incorporation of information from the database into the product model.

The general data schema is applied to develop a specific data schema for production machines. This data schema is used to create a database that
contains information about production machines. During design, this database can be applied to create the product model for production machines. Several search queries have been defined that facilitate the easy retrieval of information from the database.
7

Implementation and Case Studies

7.1 Introduction

This chapter describes the implementation of the developed models for production machines and the case studies performed with this implementation.

Section 7.2 describes the prototype design support system. This prototype was used to consult designers during the development of the models and to determine the feasibility of the models. Section 7.3 discusses three case studies that were performed by using the prototype. Section 7.4 elucidates a number of desired improvements that the cases revealed. The chapter ends with conclusions.

7.2 Prototype design support system

The developed meta product model, procedures of design activities, and data schema have been integrated into a prototype design support system called Design It. This prototype was used to consult designers during the research and to demonstrate that an integration of the meta product model, procedures of design activities, and data schema is feasible.

Consulting designers

To ensure that the developed models connect to the designers' way of thinking, experienced designers were timely involved in the research project and were confronted with ongoing results. It appeared that practical designers are less interested in theoretical models and more interested in a tangible application. A confrontation with a premature implementation resulted in very concrete responses as “maybe we should have a button to analyse the cycle-time”,
“dragging this function to a solution would be convenient” or “can we search with parameters?”. This observation fortified the importance of the implementation. The suggestions and comments on the prototype were used to modify and develop the initial models and prototype.

Feasibility
Figure 96 shows the relation between the models and the prototype-tool. The prototype-tool is implemented in Microsoft Visual Basic® and Microsoft Access®. Objects in the meta product model relate to object-classes in Visual Basic. Procedures of design activities relate to the methods of these object-classes. However, several activities can not be implemented as methods of classes and are implemented as auxiliary procedures. The data schema relates to the definitions of tables and relations in Access.

![Diagram](image)

**Figure 96** Relation between the model of the design process and the prototype

Figure 97 shows the main window of the design support system. It consists of an area displaying the alternative concepts within a project and an area displaying the contents of the views describing these concepts. The presentations used in the prototype are similar to those in the product model. A detailed description of the design support system can be found in Appendix B.
Figure 97 Main window of the prototype design support system Design It!

The prototype presents the product model of production machines to the designer by using the views as described in section 4.3: the product state view, function view, solution view, solution layout view and state transition view. Further, it contains a view that presents the information from the product state, function, and solution view in one overview.

The design activities mentioned in section 5.3 were implemented. Amongst others, the implementation supports the documentation of the product model, maintains the consistency between the views, determines the correctness of the state transition diagram, and simulates the transitions in the machine.

The implemented database uses the data schema from section 6.3 and contains, amongst others, information about solutions, functions, product states, and the relations between them. Queries can be executed to search information in the database.

The product model, design activities, and database were implemented without major problems, which indicates the feasibility of an integration of these models in a prototype.
7.3 Case studies

This section describes three case studies that have been performed to demonstrate that the prototype assists during the design of a production machine. The cases mainly focus on the product model and the design activities since the usage of the database requires a reasonable amount of stored information, which is however not yet available.

The design of production machines is often the redesign of an existing machine, which means that an existing design is modified to obtain a design of a new machine. Therefore, the cases started with a short explanation of an actual design of a machine. The explanation contained a picture of the machine and a description of its working method. It was used to create a product model of the machine according to the developed meta product model of production machines. Since the explanation of the machine mainly contained information about the geometry of the machine, the solution layout model in the product model was created rather easily. However, the other aspect models (the product state, function, solution, and state transition model) had to be designed from scratch. The obtained product model of the machine can be used to redesign the machine. However, since the development of the product model already demonstrated how the prototype and the underlying models supported the design process, an actual redesign of a machine is not performed.

Sections 7.3.1 to 7.3.3 present the cases and section 7.3.4 discusses the results.

7.3.1 Case 1: Block-packaging

Description

Figure 98 shows a block-packaging machine. Most important action in this machine is the vertical transport of a block and a rectangular piece of paper through the non-moving folding-device. Within this movement, five out of six planes are folded around the block. Other actions deal with supplying blocks and paper, folding the bottom plane and removing packed blocks.

The machine starts with pusher (1) that positions a block on vertical pusher (5), which is in its bottom position. Simultaneously, the paper is supplied by a pair of tongs (2), while catch (3) is closed. A pair of scissors (4) cuts the paper, while vertical presser (6) presses the paper on the block. After cutting, block and paper are transported through the folding device by vertical pusher (5). When the vertical pusher is in its top-position, the side-folders (7) fold both side-flaps and support the block. Since they may not collide with the vertical pusher, they must wait until the vertical pusher is lowered before they can continue their movement. The front-folder (8) moves towards the side-folders and simultaneously makes a fold and supports the block. Finally, back-folder
(9) makes the last fold by pushing the block on the gutter. Meanwhile, vertical pusher (5) moved down, so the process of supplying a block and paper has started again.

![Diagram of Block-Packaging machine]

**Figure 98** Block-Packaging machine

**Results**
The block-packaging machine has been modelled by using the prototype design support system. Figure 99 shows an excerpt of the functions, solutions, product states, and their relations. Rounded rectangles depict solutions, greyed rectangles indicate functions that transform a product-state, and hatched rectangles show functions that a solution requires realising its function. For example, the scissors realise the separate-function, but require the fix-function, which is realised by the vertical presser.
Figure 99 Excerpt of the block-packaging machine

Figure 100 shows an excerpt of the state transition diagram. Bold transitions indicate that they are connected to a function. For example, when the pusher moves from right to left, it performs the transport-function.

Figure 100 Excerpt of the state transition diagram
7.3.2 Case 2: Meat-taper

Description
In a slaughterhouse, pieces of meat are provided with tape, on which they are hanged for transport. The meat is manually put in the machine and the plastic tape is supplied through a hollow needle that can rotate around a fixed centre (Figure 101).

![Figure 101 Situation I: startpositions](image)

Catch A pulls the tape downward to the welding position. Subsequent, the needle rotates and sews the meat. Next, the tape is fixed by catch B after which the scissors cut it (Figure 102). Catch B rotates and the welding-tong moves from the back to the welding position. After the tape is welded, meat and tape are manually removed from the machine.

![Figure 102 Situations II and III: cutting and welding](image)
Results
Figure 103 show the product-states, functions and solutions of the meat-taper.

Figure 103 Product states, functions, and solutions of the meat-taper
Figure 104 shows the state transition diagram for the meat-taper.
7.3.3 Case 3: Flow-packer

Description
Figure 105 shows a flow-packer that packs dosed amounts of granular products in foil bags. When the frame moves down with closed catches, it pulls the foil over a forming-shoulder, which creates the vertical shoot. Simultaneously, the catches create the transverse seal. The vertical seal is made by heated side-sealers. The granular products are supplied when the frame moves down.

![Flow-Packer Diagram](image)

**Figure 105 Flow-Packer**

Results
Figure 106 shows the state transition diagram for the flow-packer.

![State Transition Diagram](image)

**Figure 106 State transition diagram for the flow-packer**
7.3.4 Discussion

The results of the cases are very promising. The machines were easily and rapidly modelled by using the prototype design support system.

The obtained product models contain both geometric and non-geometric information and give a comprehensive understanding of the machine. Especially the state transition diagram is valuable since it contains information about solutions, their movements, functions, product states, and the relationships between these items. It contributes to a better understanding and insight of the complex transition pattern of solutions and the factors (e.g. collisions and sequences) that determine these transitions. The function view gives the designer an abstract view on the machine, which could stimulate creativity. The product state view clearly shows the stages the product has during production. The solution and solution layout view show how different solutions co-operate and how their functioning depends on the functioning of other solutions.

The implemented design activities are very convenient. Although many activities are automated, they are subordinate to the designer and therefore they do hardly obstruct him in his working method. Automation of administration activities significantly decreased design time. The automatic maintenance of the consistency between views has the advantages that designers do not have to spend time and attention doing this, and that views are always consistent. From the synthesis activities, the automatic elimination of insignificant steps and auto modification were mainly used since other synthesis activities required a reasonable filled database. The elimination of insignificant steps appeared to be very profitable in decreasing the number of steps in the state transition diagram. Auto modify was very convenient since it took over activities the designer normally had to perform. However, occasionally unexpected results were obtained, which nevertheless could easily be resolved. Analysis activities turned out to be very valuable. Demands in the state transition diagram were checked automatically and continuously. Consequently, the designer always had an up-to-date description of the correctness of the state transition diagram, which supported him in rapidly correcting possible errors and in optimising the cycle time. Simulating the transitions of the solutions in the solution layout view revealed overlooked collision and concurrent-demands, and contributed to a reduction of the number of errors in the final design.

The database was mainly used to select functions. The structure of the database allowed the selection of functions with different levels of abstraction. For example, the function of the scissors can be "cut", but at a higher level of abstraction, the function "separate" could be chosen. This function is more general and can yield unexpected solutions. Although the number of items in
the database was small, the cases showed the possibility to reuse functions, solutions, product states, and their relationships.

7.4 Desired improvements

The cases revealed a number of desired improvements. It must be emphasised that in many cases, these improvements relate to the prototype and especially to the user-interface of the prototype, and not to the developed models. Adding these improvements is mainly a matter of implementing.

7.4.1 Product model

Supplementary views

Although the product model gave a comprehensive understanding of a production machine, two supplementary views can increase understanding: a continuous state transition view and a machine cycle view.

The state transition diagram shows the transitions of the solutions by using the time in steps and textual properties. However, a continuous state transition diagram should show the transitions by using the time in seconds and numerical properties (Figure 107). Besides, the exact course of the transition in time should be presented. The extra information about properties and time can be used to perform various calculations (e.g. accelerations and required power of a motor) in design-stages latter to the conceptual design stage.

![State Transition Diagram](image)

![Continuous State Transition Diagram](image)

**Figure 107** A state transition and a continuous state transition diagram
The state transition diagram shows the transitions within the machine from the product point-of-view. Another additional view should show the transitions from a machine point-of-view. To explain this *machine cycle view*, an imaginary machine consisting of a transporter, and forming devices 1 and 2 is used (Figure 108). When the transporter moves from left to right, it moves three boxes. When it moves from right to left, the catches on the transporter rotate and the boxes keep their positions. The two devices 1 and 2 sequentially process the boxes.

![Figure 108 Example to explain the machine-cycle](image)

The state transition diagram (Figure 109) shows the transitions of the solutions, but also the transitions of the product, which are indicated by bold transitions and arrows. The figure shows that it takes five steps to manufacture the product. Device 2 requires step 6 to return to its initial position.

![Figure 109 State transition diagram of the example](image)

When the example is viewed from a machine point-of-view, the state transition diagram as shown in Figure 110 is obtained. This diagram shows that the cycle-time of the machine is three steps, which means that every three steps a product is finished. This in contrary to the product-cycle time, which defines the time required to manufacture one product.
Figure 110 Machine-cycle of the example

Figure 111 shows the relation between the state transition and machine-cycle diagram. Within this figure, multiple machine-cycles are concatenated, which makes it possible to draw the product-cycles of products i and i+1. For clarity, the cycles of products i-1 and i+2 are omitted.

Figure 111 Relation between machine- and product-cycles

A machine cycle view supports the designer to determine and optimise the number of products a machine produces per minute. Similar to the state transition diagram, which is supplemented by a continuous state transition diagram, the machine-cycle can be described in both discrete and continuous time-domain.
Improved solution layout view
The prototype design support system presents the solutions in the solution layout view as plain rectangles. The transitions of the solutions are visualised by changing the dimensions of the rectangles. Although these simple presentations already give the designer an idea about the configuration of the machine, presentations that are more realistic are desired, for example 3D-models of the solutions.

Another improvement to the solution layout view can be the visualisation of product states. Current solution layout view only presents the solutions and does not show the products in process. If product states are presented as well, the interactions between solutions and product states can become more clearly.

Additional state transition profiles
Within current product model, the transition of a solution is described by properties, states, and transitions. With these elements, an unlimited number of transition profiles can be created. Figure 112 shows two examples of very common and implemented profiles.

![Diagram](image)

**Figure 112 Examples of common state transition profiles**

However, the cases made clear that additional profiles are desired. Figure 100 showed that the folding-device, although it does not move, was designated a transition, because a functional-transition was needed to create and show the sequential- and concurrent-demands. A state transition profile as shown in Figure 113 would be more convenient. This profile can be obtained by creating three states that refer to the same property. However, current prototype does not offer this flexibility. It must be emphasised that this is pure a limitation of the implementation and not a limitation of the proposed meta product model.
Figure 113 Desired additional state transition profile I

Figure 106 showed that, similar to the folding-device, the side-sealer was designated a transition, while it has an on/off-character. The state transition profile shown in Figure 114 would be more realistic. Again, the meta product model does support this state transition profile since it consists of properties, states and transitions, but the specific presentation is lacking in the prototype.

Figure 114 Desired additional state transition profile II

Figure 115 shows another state transition profile that can be described by the meta product model, but that the prototype does not support.

Figure 115 Desired additional state transition profile III

If these additional state transition profiles are added to the prototype, the state transition diagram will be more realistic.

7.4.2 Design activities

Correction of state transition diagram

Although the prototype gives visual signals when a demand within the state transition diagram is incorrect, it is desired to have a design activity that automatically solves this incorrectness. For example, the sequential demand between transitions AB and BC (Figure 116) can be corrected by moving state B to the left, moving state C to right, or by a combination of both. The correction activity should decide which of the possible corrections is preferred and should correct concurrent demands and collision indicators as well.
Detection of collisions
The cases showed that product models contain a large number of collision indicators, which manually have to be added. However, these collision indicators can often be extracted from the solution layout view. Figure 117 shows an example with two cylinders. Based on their mutual positions and their stroke, it can be determined in which situations they never (✓), always (×), or possible (?) collide. This information can be used to automatically generate collision indicators between transitions of these solutions.

Decomposition
The cases resulted in reasonably large product models. Decomposition can be used to divide these complex product models into smaller and less complicated sub product models. It must be investigated how the meta product model can describe decomposition and how a decomposition/composition activity is described. A possible difficulty is that, since decomposition can take place in
each view, the consistency between decompositions in different views must be managed. Another problem is the maintenance of the relations between sub-product models and between child- and parent product models.

7.4.3 Database
The used database contained only a few items. Nevertheless, it showed that information can be reused. However, it must be investigated how increasing the number of items in the database affects reusing information. For example, when the database is large, search algorithms can return a number of items that designers can not oversee. Additional filtering of information could be required to select between possible items. It must be determined which criteria can be used to filter the items. A possibility is to add a specification view to the meta product model and rank found items according to their performance on the specifications within this view. Other issues to investigate are filling and maintaining the database.

7.5 Conclusions
This chapter described the implementation of the developed models for production machines and the case studies performed with this implementation.

The meta product model, procedures of design activities, and data schema of production machines have been implemented in a prototype design support system. This implementation was used to consult designers during the research. The comments of the designers were used to modify and improve the prototype and the models, which connects them to the designers’ way of thinking. Further, the implementation demonstrates that the models can be integrated into a design support system.

The cases use the prototype to demonstrate that actual designs of production machines can easily be modelled according to proposed meta product model by using the procedures of design activities and by consulting the database.

The created product models give a comprehensive understanding of diverse aspects of the machine. Especially the state transition model turned out to be valuable in describing the machine.

The automated design activities save time and reduce the errors in the created product model. Especially the activities that check the demands in the state transition diagram are valuable since they assist in creating a correct design of the machine. Although many activities are automated, they do not obstruct the designer in his working.
Although the database contained only a limited amount of information, it demonstrates that it can be consulted during the design process. It was mainly used to retrieve functions of the machine.

The cases revealed several desired improvements to the prototype, product model, design activities, and database. These improvements mainly concerned the prototype. Current prototype contributed to develop the models and to perform case studies, but it is not mature enough to be evaluated in practice. Therefore, the prototype and models require further development and evaluation.
Conclusions

8.1 Introduction

This chapter contains the conclusions of this research. First, the research goals are reviewed and it is determined whether they have been achieved. Second, the deliverables of the research are summarised. Third, recommendations for future research are made.

8.2 Review of the research goals

The goal of this thesis has been defined as (section 1.3): to investigate how designers of production machines can be supported during the conceptual stage of the design process.

To achieve this main research goal, four sub-goals were defined:

- to review the state of the art in design research,
- to investigate how designers can be supported during conceptual design,
- to develop a computer-based approach to support the design process, and
- to evaluate the approach by implementing it in a prototype design support system.

To reach these goals, a hypothesis about the design process was introduced. This hypothesis states that a product model, design activities, and a database can model the conceptual design process. The product model describes the design in progress, design activities define the operations that are executed to create this design, and the database contains general background information that can be used in this design.
Review of current design research
Existing design models, design theories, and computer-based approaches that support the design process were reviewed (Chapter 2). This investigation gained insight in the design process and identified the deficiencies and opportunities of the approaches. The main deficiencies are that the models are very academic and general, which makes it difficult to apply the models directly to the design of production machines, and that applicable tools are absent. The main opportunities are that the general models can serve as a basis to develop specific models for production machines and that several taxonomies and catalogues are available to create the database.

The state of the art showed that the elements of the hypothesis are found in many models and methods, but that often only one or two elements of the hypothesis are modelled. Besides, a plain and conscious distinction between the elements is missing.

Investigation of how designers can be supported in the conceptual design stage
It has been investigated which elements are necessary to model the conceptual design process, which resulted in the derivation of the hypothesis (Chapter 3).

The elements of the model of the design process were used to indicate specific shortcomings of current conceptual design stage. The product model consists of multiple aspect models, which makes it difficult to maintain the consistency between these aspect models. This is fortified by the absence of a uniform and formal definition of the aspect models. The design activities can be very monotonous and time-consuming, which can lead to errors. The database often contains only geometric information and retrieved information can not easily be incorporated into the product model.

It was argued that each of the elements had to be supported by the pursued design support system. Therefore, a meta product model, procedures of design activities and a data schema have been introduced. The meta product model is used to instantiate an actual product model, procedures of activities are used to execute actual design activities, and the data schema is applied to create an actual database.

Development of a computer-based approach to support conceptual design
A general computer based description of the meta product model (Chapter 4), procedures of design activities (Chapter 5), and data schema (Chapter 6) has been developed. These general descriptions were used to create specific models for production machines.
Evaluation of the approach by implementing it into a working prototype
The models were integrated and implemented in a prototype design support system and several cases have been performed by using this prototype (Chapter 7). The implementation and cases show the feasibility of the approach and that conceptual design process can be supported by proposed design system.

8.3 Deliverables

This section reviews the deliverables of the research: the model of the design process, a meta product model, procedures of design activities, a data schema, the prototype design support system.

8.3.1 Model of the design process
Chapter 3 described a general model of the design process. This model describes the conceptual design process by a product model, design activities, and database. A meta product model supports the creation of the product model, models of design activities support performing design activities, and a data schema supports the creation of a database.

8.3.2 Meta product model
Chapter 4 described a general meta product model that has been used to develop a specific meta product model for production machines. The general meta product model uses multiple aspect models to describe the design. Each of these aspect models highlights the product to be designed from a particular point of view. The meta product model consists of formal definitions of the data schema, multiple views, and mapping rules. The data schema defines the structure of the data in the product model, the views present this data to the designer, and the mapping rules define how the data relates to the presentations in the views. A method to add views to the meta product model has been developed, which makes it possible to extend the meta product model in future.

The meta product model for production machines consists of five aspect models: a product state-, function-, solution-, solution layout-, and state transition model. The product state- and function model describe the states that the product to be produced has during production, the functions that transform these states, and the sequence of the transformations. The solution- and solution layout model describe the initial geometry and spatial positions of the solutions that realise these transformations. The state transition model shows which transition of a solution realises a transformation and how the transitions of different solution relate to each other.
The meta product model for production machines gives a comprehensive understanding of the design of the production machine. The data schema guarantees uniform data, the views unambiguously present this data, and the mapping rules define the consistency between data and views.

8.3.3 Procedures of design activities
Chapter 5 described general design activities that a designer performs. These activities are grouped into five categories: administration-, presentation-, synthesis-, analysis-, and evaluation-activities. Procedures of these activities are developed in such manner that they can be supported or automated by a design support system. The procedures prescribe which input an activity requires, the performed operations, and what the resulting output will be. The procedures serve as bases to support or automate design activities.

The general procedures have been applied to develop a number of specific procedures of design activities for production machines. The administration- and presentation-activities support during the creation and visualisation of the design of the production machine. The synthesis activities assist in retrieving information from the database and in optimising the design. The analysis activities check the correctness of the design and can provide a simulation of the transitions within the design. Evaluation activities are not yet supported.

The design activities are supported and in several cases automated. This results in reducing the errors in the design and the time wasted on monotonous activities.

8.3.4 Data schema
Chapter 6 described how common database techniques were used to describe a general data schema. This data schema can be used to create a database with background information that can be incorporated into the product model.

The general data schema has been used to develop a specific data schema for production machines. This data schema is based on the meta product model for production machines and supplemented with categories to make the database more accessible. Queries can be executed to retrieve information from the database.

The developed data schema defines the structure of the database, which ensures a uniform storage. Since it is based on the meta product model, the data schema enables the storage and retrieval of both geometric and non-geometric information. Further, it enables the effortless incorporation of the retrieved information in the product model.
8.3.5 Prototype design support system
The meta product model, design activities-models, and data schema have been implemented in a prototype design support system. Chapter 7 describes how this prototype has been used to consult designers and to perform case studies. The implementation shows the feasibility of the model of the design process, and demonstrates that an actual design can be modelled by proposed design support system.

8.4 Recommendations for future research
The prototype design support system and the performed cases showed a number of desired improvements to the models and the prototype. The meta product model can be improvement by adding the continuous state transition diagram and by the model that presents the machine cycle. The procedures of design activities should focus on the automatic correction of these diagrams, since that can save much time. The database contains information about designs. It is recommended to investigate how a database can contain information about the design process and how this can be used to perform the design process. Further, the user-interface and the user-friendliness of the prototype should be improved.

During the cases, the database contained only a limited number of items. It is recommended to extend the number of items and to investigate how this affects the reuse of information from the database.

The models focus on the conceptual design stage. It must be investigated how they connect to the stages before and after conceptual design.

It is recommended to evaluate the models more thoroughly in practice. However, before this evaluation can be performed, a more mature prototype and suitable cases must be available.

Current prototype design support system works stand-alone. To use the system in practice, it must connect to commercial CAD-tools the companies currently use. Therefore, it must be investigated how the prototype can connect to these tools.
Bibliography


Klein Breteler, A.J. (1987), Kinematic optimization of mechanisms, a finite element approach, Thesis Delft University of Technology, Delft


153
Appendix A

Formal Description

A.1 Glossary

The following abbreviations are used to indicate the data schemas, objects, and collections:

- **CD**: collection of concurrent demands
- **cd**: concurrent demand
- **CI**: collection of collision indicators
- **ci**: collision indicator
- **ds**: data schema of the meta product model
- **FR**: collection of functional relations
- **fr**: functional relation
- **fu**: function
- **FU**: collection of functions
- **fuv**: data schema of the function model
- **ps**: product state
- **PS**: collection of product states
- **psm**: data schema of the product state model
- **PSP**: collection of product state properties
- **psp**: product state property
- **RFRST**: collection of relations between a functional relation and solution state transition
- **rfrst**: relation between a functional relation and solution state transition
- **RR**: collection of requires relations
- **rr**: requires relation
- **SD**: collection of sequential demands
- **sd**: sequential demand
- **slm**: data schema of the solution layout model
- **so**: solution
- **SO**: collection of solutions
- **som**: data schema of the solution model
A.2 Data schemas of the aspect models

This section presents the data schemas of the aspect models. The collections, the objects, and the restriction on the attributes are given.

<table>
<thead>
<tr>
<th>Product state model</th>
</tr>
</thead>
<tbody>
<tr>
<td>$psm$ = $\langle PS, TRPSPS \rangle$</td>
</tr>
<tr>
<td>$PS$ = { $ps_i$ }</td>
</tr>
<tr>
<td>$TRPSPS$ = { $trpsps_k$ }</td>
</tr>
<tr>
<td>$ps_i$ = $\langle name, picture, PSP \rangle$</td>
</tr>
<tr>
<td>$PSP$ = { $sp_j$ }</td>
</tr>
<tr>
<td>$sp_j$ = $\langle name, picture, value \rangle$</td>
</tr>
<tr>
<td>$trpsps_k$ = $\langle ps_{org}, ps_{dist} \rangle$</td>
</tr>
</tbody>
</table>

\[ ps_{org} = \text{origin} \]
\[ ps_{org} \in PS \]
\[ ps_{dist} \in PS \]
\[ ps_{org} \neq ps_{dist} \]
### Function model

\[
\begin{align*}
\text{fun} &= \langle \text{FU, TRFUFU, RR} \rangle \\
\text{FU} &= \{ \text{fu} \} \\
\text{TRFUFU} &= \{ \text{trfu} \} \\
\text{RR} &= \{ \text{rr} \} \\
\text{fu} &= \langle \text{name} \rangle \\
\text{trfu} &= \langle \text{fu}_\text{org}, \text{fu}_\text{dat} \rangle \\
\text{rr} &= \langle \text{fu}_\text{org}, \text{fu}_\text{dat} \rangle
\end{align*}
\]

- \( \text{fu}_\text{org} = \text{origin} \)
- \( \text{fu}_\text{org} \in \text{FU} \)
- \( \text{fu}_\text{dat} \in \text{FU} \)
- \( \text{fu}_\text{org} \neq \text{fu}_\text{dat} \)

### Solution model

\[
\begin{align*}
\text{som} &= \langle \text{SO, TRSOSO, RR} \rangle \\
\text{SO} &= \{ \text{so} \} \\
\text{TRSOSO} &= \{ \text{trsoso} \} \\
\text{RR} &= \{ \text{rr} \} \\
\text{so} &= \langle \text{name, picture} \rangle \\
\text{trsoso} &= \langle \text{so}_\text{org}, \text{so}_\text{dat} \rangle \\
\text{rr} &= \langle \text{so}_\text{org}, \text{so}_\text{dat} \rangle
\end{align*}
\]

- \( \text{so}_\text{org} = \text{origin} \)
- \( \text{so}_\text{org} \in \text{SO} \)
- \( \text{so}_\text{dat} \in \text{SO} \)
- \( \text{so}_\text{org} \neq \text{so}_\text{dat} \)

- \( \text{so}_\text{org} = \text{origin} \)
- \( \text{so}_\text{org} \in \text{SO} \)
- \( \text{so}_\text{dat} \in \text{SO} \)
- \( \text{so}_\text{org} \neq \text{so}_\text{dat} \)
### Solution layout model

\[
\begin{align*}
stm & = \{SO\} \\
SO & = \{so_j\} \\
so_j & = \{x_{org}, y_{org}, \beta_{org}, length, height\}
\end{align*}
\]

### State transition model

\[
\begin{align*}
stm & = \{SO, SD, CD, CI, cycle\_time\} \\
SO & = \{so_j\} \\
SD & = \{sd_k\} \\
CD & = \{cd_i\} \\
CI & = \{ci_t\} \\
so_j & = \{name, SP_j, SS_j, ST_j\} \\
SP_j & = \{sp_i\} \\
SS_j & = \{ss_m\} \\
ST_j & = \{st_n\} \\
sp_i & = \{name\} \\
ss_m & = \{sp_j, stepnumber\} \quad sp_j \in SP_j, 0 \leq stepnumber < stm.\_cycle\_time, stepnumber < \text{cycle}\_time \\
ss_\text{org} & = \text{origin} \\
ss_\text{org} & \in SS_j \\
st_n & = \{ss_\text{org}, ss_\text{dst}, type\} \\
ss_\text{dst} & \in SS_j \\
\text{type} & \in \{\text{function, no}\_\text{function}\} \\
ss_\text{org} & \neq ss_\text{dst} \\

sd_k & = \{st_\text{org}, st_\text{dst}, correctness\} \\
st_\text{org} & = \text{origin} \\
st_\text{org} & \in so_\text{org}.ST \\
st_\text{dst} & \in so_\text{dst}.ST \\
so_\text{dst} & \in SO \\
so_\text{org} & \neq so_\text{dst} \\
\text{correctness} & \in \{\text{incorrect, optimal, correct}\}
\end{align*}
\]
\[ cd_k = \langle st_{org}, st_{dist}, correctness \rangle \]
\[
\begin{align*}
& st_{org} \neq st_{dist} \\
& st_{org} = \text{origin} \\
& st_{org} \in so_{org} \cdot ST \\
& st_{dist} \in so_{dist} \cdot ST \\
& so_{dist} \in SO \\
& so_{org} \in SO \\
& so_{org} \neq so_{dist} \\
& correctness \in [true, false]
\end{align*}
\]

\[ ci_k = \langle st_i, st_m, correctness \rangle \]
\[
\begin{align*}
& st_i \neq st_m \\
& st_i \in so_i \cdot ST \\
& st_m \in so_m \cdot ST \\
& so_i \in SO \\
& so_m \in SO \\
& so_i \neq so_m \\
& correctness \in [true, false]
\end{align*}
\]
A.3 Data schema of the meta product model

This section presents the data schemas of the meta product model. The collections, the objects, and the restriction on the attributes are given.

<table>
<thead>
<tr>
<th>Data schema of the meta product model</th>
</tr>
</thead>
<tbody>
<tr>
<td>( ds ) = { FU, SO, PS, TRFUPS, TRSOPS, FR, RFRST, RR, CI, cycle_time }</td>
</tr>
<tr>
<td>( FU ) = { fu_i }</td>
</tr>
<tr>
<td>( SO ) = { so_j }</td>
</tr>
<tr>
<td>( PS ) = { ps_j }</td>
</tr>
<tr>
<td>( TRFUPS ) = { trfups_j }</td>
</tr>
<tr>
<td>( TRSOPS ) = { trsops_j }</td>
</tr>
<tr>
<td>( FR ) = { fr_j }</td>
</tr>
<tr>
<td>( RFRST ) = { rfrst_j }</td>
</tr>
<tr>
<td>( RR ) = { rr_j }</td>
</tr>
<tr>
<td>( CI ) = { ci_j }</td>
</tr>
<tr>
<td>( fu_i ) = { name }</td>
</tr>
<tr>
<td>( so_j ) = { name, picture, x_{org}, y_{org}, b_{org}, length, height, SP_j, SS_j, ST_j }</td>
</tr>
<tr>
<td>( SP_j ) = { sp_j }</td>
</tr>
<tr>
<td>( sp_j ) = { name }</td>
</tr>
<tr>
<td>( SS_j ) = { ss_j }</td>
</tr>
<tr>
<td>( ss_j ) = { sp_j, stepnumber } \quad sp_j \in SP_j \quad 0 \leq stepnumber \quad stepnumber &lt; ds_cycle_time }</td>
</tr>
<tr>
<td>( ST_j ) = { st_j }</td>
</tr>
<tr>
<td>( st_j ) = { ss_{org}, ss_{dst} } \quad ss_{org} = origin \quad ss_{org} \in SS_j \quad ss_{dst} \in SS_j \quad ss_{org} \neq ss_{dst} }</td>
</tr>
<tr>
<td>( ps_j ) = { name, picture, correctness, PSP } \quad correctness \in { incorrect, optimal, correct }</td>
</tr>
<tr>
<td>( PSP ) = { psp_j }</td>
</tr>
<tr>
<td>( psp_j ) = { name, picture, value }</td>
</tr>
<tr>
<td>Symbol</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>$trfups_j$</td>
</tr>
<tr>
<td>$trsops_j$</td>
</tr>
<tr>
<td>$fr_j$</td>
</tr>
<tr>
<td>$rfrst_j$</td>
</tr>
<tr>
<td>$rr_j$</td>
</tr>
<tr>
<td>$ci_k$</td>
</tr>
</tbody>
</table>

- $origin \in \{"Function","State"\}$
- $fu_i \in FU$
- $ps_m \in PS$
- $so_i \in SO$
- $ps_m \in PS$
- $fu_i \in FU$
- $so_m \in SO$
- $fr_i \in FR$
- $st_m \in fr_i \cdot so.ST$
- $fr_{org} \in FR$
- $fu_{dst} \in FU$
- $fr_{org} \cdot fu \neq fu_{dst}$
- $correctness \in [true, false]$
A.4 Mapping rules

This section presents the mapping rules between the meta product model and the aspect models. The mapping rules and the restricting conditions on the involved objects are given.

Mapping between the meta product model and the product state model

\[
\langle ps_i \rangle_{pm} \quad \leftrightarrow \quad \langle ps_i \rangle_{mpm}
\]

\[
\langle trpsps_m \rangle_{pm} \quad \leftrightarrow \quad \langle trfups_i, fu_j, trfups_k \rangle_{mpm}
\]

\[
trfups_i \cdot origin = "State"
\]

\[
trfups_i \cdot fu = fu_j
\]

\[
\langle trpsps_m, ps_{ext} \rangle_{pm} \quad \leftrightarrow \quad \langle trfups_i \cdot ps \rangle_{mpm}
\]

\[
trfups_i \cdot origin = "Function"
\]

\[
trfups_k \cdot fu = fu_j
\]

\[
\langle trpsps_m, ps_{int} \rangle_{pm} \quad \leftrightarrow \quad \langle trfups_k \cdot ps \rangle_{mpm}
\]

\[
trops_i \cdot origin = "State"
\]

\[
trops_i \cdot so = so_j
\]

\[
\langle trpsps_m, ps_{ort} \rangle_{pm} \quad \leftrightarrow \quad \langle trops_i \cdot ps \rangle_{mpm}
\]

\[
trops_i \cdot origin = "Solution"
\]

\[
trops_k \cdot so = so_j
\]

\[
\langle trpsps_m, ps_{aort} \rangle_{pm} \quad \leftrightarrow \quad \langle trops_k \cdot ps \rangle_{mpm}
\]

Mapping between the meta product model and the function model

\[
\langle fu_i \rangle_{fm} \quad \leftrightarrow \quad \langle fu_i \rangle_{mpm}
\]

\[
\langle trfufu_m \rangle_{fm} \quad \leftrightarrow \quad \langle trfups_i, ps_j, trfups_k \rangle_{mpm}
\]

\[
trfups_i \cdot origin = "Function"
\]

\[
trfups_i \cdot ps = ps_j
\]

\[
\langle trfufu_m \cdot fu_{ur} \rangle_{fm} \quad \leftrightarrow \quad \langle trfups_i \cdot fu \rangle_{mpm}
\]

\[
trfups_i \cdot origin = "State"
\]

\[
trfups_k \cdot ps = ps_j
\]

\[
\langle trfufu_m \cdot fu_{dr} \rangle_{fm} \quad \leftrightarrow \quad \langle trfups_k \cdot fu \rangle_{mpm}
\]

\[
\langle rr_m \rangle_{fm} \quad \leftrightarrow \quad \langle fr_i, rr_j \rangle_{mpm}
\]

\[
rr_m \cdot fu_{org} \quad \leftrightarrow \quad \langle fr_i \cdot fu \rangle_{mpm}
\]

\[
fr_i = rr_j \cdot fr
\]

\[
\langle rr_m \cdot fu_{dist} \rangle_{fm} \quad \leftrightarrow \quad \langle rr_j \cdot fu \rangle_{mpm}
\]
Mapping between the meta product model and the solution model

\[ \langle so_j \rangle_{som} \leftrightarrow \langle so_i \rangle_{mpm} \]

\[ \langle trsoso_m \rangle_{som} \leftrightarrow \langle trsops_i \cdot ps_j \cdot trsops_k \rangle_{mpm} \]

\[ \langle rr_m \rangle_{som} \leftrightarrow \langle fr_i \cdot rr_j \cdot fu_k \cdot fr_l \rangle_{mpm} \]

Mapping between the meta product model and the solution layout model

\[ \langle so_j \rangle_{slm} \leftrightarrow \langle so_i \rangle_{mpm} \]

Mapping between the meta product model and the state transition model

\[ \langle so_j \rangle_{stm} \leftrightarrow \langle so_i \rangle_{mpm} \]

\[ \langle sd_i \rangle_{stom} \leftrightarrow \langle rfrst_{org} \cdot st \rangle_{mpm} \]

\[ \langle sd_i \cdot st_{org} \rangle_{stom} \leftrightarrow \langle rfrst_{org} \cdot st \rangle_{mpm} \]

\[ \langle fr_{org} \cdot fr = fr_{org} \rangle \]

\[ \langle fr_{org} \cdot fu = fu_{org} \rangle \]

\[ \langle trfups_{org} \cdot fu = fu_{org} \rangle \]

\[ \langle trfups_{org} \cdot origin = "Function" \rangle \]

\[ \langle trfups_{org} \cdot ps = ps_i \rangle \]

\[ \langle trfups_{org} \cdot ps = ps_i \rangle \]

\[ \langle trfups_{org} \cdot origin = "State" \rangle \]

\[ \langle trfups_{org} \cdot ps = ps_i \rangle \]

\[ \langle fr_{dist} : fu = fu_{dist} \rangle \]

\[ \langle fr_{dist} : fu = fu_{dist} \rangle \]

\[ \langle fr_{dist} : fr = fr_{dist} \rangle \]

\[ \langle sd_i,s_{dist} \rangle_{stom} \leftrightarrow \langle rfrst_{dist} \cdot st \rangle_{mpm} \]
\( \langle c_i \rangle^{\text{sim}} \iff \langle \text{sim} \rangle^{\text{mpm}} \)

\[
\begin{align*}
\langle \text{cd} \rangle^{\text{sim}} & \iff \langle \text{sim} \rangle^{\text{mpm}} \\
\text{rfrst}_{\text{org}}^{\text{dc}} \cdot \text{fr}_{\text{org}}^{\text{dc}} & = \text{rr}_{i, \text{dc}}^{\text{dc}} \\
\text{fu}_{\text{dist}}^{\text{dc}} \cdot \text{fr}_{\text{dist}}^{\text{dc}} \cdot \text{rfrst}_{\text{dist}}^{\text{dc}} & = \text{rr}_{i, \text{dist}}^{\text{dc}} \\
\text{fu}_{\text{dist}}^{\text{dc}} \cdot \text{fu}_{\text{org}}^{\text{dc}} & = \text{fu}_{\text{org}}^{\text{dc}} \\
\text{fr}_{\text{dist}}^{\text{dc}} \cdot \text{fu}_{\text{dist}}^{\text{dc}} & = \text{fu}_{\text{dist}}^{\text{dc}} \\
\text{rfrst}_{\text{dist}}^{\text{dc}} \cdot \text{st} = \text{st}_{\text{dist}} & \iff \langle \text{cd} \rangle^{\text{sim}} \iff \langle \text{sim} \rangle^{\text{mpm}} \\
\text{rfrst}_{\text{org}}^{\text{dc}} \cdot \text{fr} = \text{fr}_{\text{org}}^{\text{dc}} & = \text{rr}_{i, \text{dist}}^{\text{dc}} \\
\text{rr}_{i, \text{dist}}^{\text{dc}} \cdot \text{fu}_{\text{org}}^{\text{dc}} & = \text{fu}_{\text{org}}^{\text{dc}} \\
\text{rr}_{i, \text{dist}}^{\text{dc}} \cdot \text{fu}_{\text{dist}}^{\text{dc}} & = \text{fu}_{\text{dist}}^{\text{dc}} \\
\text{fr}_{\text{dist}}^{\text{dc}} \cdot \text{fu} = \text{fu}_{\text{dist}}^{\text{dc}} & = \text{rr}_{i, \text{dist}}^{\text{dc}} \\
\text{rfrst}_{\text{dist}}^{\text{dc}} \cdot \text{st} = \text{st}_{\text{dist}} & = \text{rr}_{i, \text{dist}}^{\text{dc}} \\
\text{rfrst}_{\text{dist}}^{\text{dc}} \cdot \text{st} & = \text{st}_{\text{dist}}
\end{align*}
\]
Appendix B

Prototype
Design support system

This section presents a number of screen-shots of the prototype design support.

B.1 Main window

Figure 118 shows the main window of the system. The main window consists of a window that displays the concepts within a project and a window that displays the contents of the views that describe these concepts.

![Main Window of the design support system](image)

**Figure 118** Main window of the design support system
B.2 Views

By selecting "tabs", a specific view is displayed. Figure 120 to Figure 123 show examples of the contents of each view.

Figure 119 Example of the product state view
Figure 120 Example of the function view

Figure 121 Example of the solution view
Figure 122 Example of the state transition view

Figure 123 Example of the solution layout view
B.3 Browsing the database

Functions, product states, and solutions can directly be selected from the database by using the interfaces shown in Figure 124 to Figure 126.

Figure 124 Selecting a function from the database

Figure 125 Selecting a product state from the database
B.4 Searching the database

Search-activities result in a set of objects that can be used in the product model. Figure 127 shows an example of the output of a search-routine that searches for solutions that can realise an input-function.

Figure 127 Results after searching a solution that can realise the function "transport"
Curriculum Vitae

Barry Hendrik de Roode was born on October 3, 1972, in Vlaardingen, the Netherlands. In 1990, he finished primary education at the "Christelijke Scholengemeenschap Westland-Zuid" and started his study at the Faculty of Mechanical Engineering at Delft University of Technology. He specialised in the field of Mechanisation of Production at the Laboratory of Production Engineering and Industrial Organisation. Under supervision of Prof. ir. H.A. Crone, he finished his study in 1995 with a Master's degree on the subject "Simulation of the system servo-drive and mechanism". In 1995, he started at the same laboratory as a research assistant in the area of computer supported conceptual design of production machines, resulting in this thesis.
Computer Supported Design of Production Machines

This thesis describes the development of a design support system for the design of production machines. The design support system implements a model of the design process that consists of three elements: a product model, design activities, and a database.

The product model describes the design of the production machine and consists of multiple aspect models, each highlighting the design from a particular point of view. The following aspect models have been developed: a product state-, function-, solution-, solution layout-, and state transition model.

The design activities are operations that modify the product model. They are divided in five categories: administration, presentation, synthesis, analysis, and evaluation.

The database contains information about past designs that can be reused in a new design. Several search queries have been developed that facilitate the easy retrieval of information from the database.

The design support system integrates and implements the developed models and shows their feasibility. It was successfully used to perform a number of case studies in which production machines were modelled by using the system. The results of these cases are promising and indicate that the design support system supports the design of production machines.