FunKey Architecting

An Integrated Approach to System Architecting Using Functions, Key Drivers and System Budgets

G. Maarten Bonnema

FUNKEY ARCHITECTING

AN INTEGRATED APPROACH TO SYSTEM ARCHITECTING USING FUNCTIONS, KEY DRIVERS AND SYSTEM BUDGETS

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
Prof. Dr. W.H.M. Zijm,
volgens het besluit van het College voor Promoties
in het openbaar te verdedigen
op donderdag 3 april 2008 om 15.00 uur

door

Gerrit Maarten Bonnema
geboren op 23 april 1968
te Hengelo (Overijssel)

Dit proefschrift is goedgekeurd door de promotor Prof. Dr. Ir. F.J.A.M. van Houten.

FunKey Architecting

an Integrated Approach to System Architecting Using Functions, Key Drivers and System Budgets

PHD THESIS

By Gerrit Maarten Bonnema at the Department of Engineering Technology (CTW) of the University of Twente Enschede, the Netherlands.

Enschede, 3 april 2008

De promotiecommissie:

Prof. Dr. F. Eising	Universiteit Twente	voorzitter, secretaris
Prof. Dr. Ir. F.J.A.M. van Houten	Universiteit Twente	promotor
Prof. Dr. Ir. Ing. A.G. Dorée	Universiteit Twente	
Prof. Dr. Ir. J. van Amerongen	Universiteit Twente	
Prof. Dr. A.H. Basson	Stellenbosch University, Zuid Afrika	
Prof. Dr. G.J. Muller	Buskerud University College, Noorwegen; Embedded Systems Institute, Eindhoven	
Prof. Dr. T. Tomiyama	Technische Universiteit Delft	
Dr. Ir. D.A. Schipper	Demcon	

Keywords: System design, system architecting, system budgets, key drivers, complex systems, multidisciplinary research

ISBN: 978–90–365–2631–9

Cover image by Maaike Nijkamp
Cover design by G. Maarten Bonnema
Printed by PrintPartners Ipskamp, Enschede
This work has been published using LaTeX 2ε and the Twentethesis documentclass.

Voor mijn vader ☥

Summary

Present state of the art products consist of integrated electronic, mechanical and software modules. Integration is present both within and between modules; even more so for complex systems like wafer steppers, aircraft, and medical imaging systems. Designing such systems requires close cooperation of electrical, mechanical and software engineers. However, these engineers speak different languages, and are presently ever more spread around the world. These specialist engineers often have a view of their own task only. The computer tools they use stimulate the build-up of the design in a bottom-up approach (from detail to assembly to system). Therefore the context information the specialists have is too limited or even lacking. The system designer, on the other hand, lacks the detail information needed to supervise the system integrity during the entire process. In this thesis a method is developed that helps the designer to acquire context information, and the system designer to track the detail information. The goal is threefold: achieve *insight*, create and maintain *overview*, and stimulate *innovation*.

In the first part of the thesis, the process of designing complex systems is investigated. First the design process is investigated. The views of different authors are presented. The common view is that the process is to be split in specification and requirement development; conceptual design; embodiment design; and detail design. The system architecture that is the backbone of the design is developed in the conceptual phase. Designers working in that phase have been interviewed to determine what product models they use. The result is that in the early phase of complex design projects, system budgets; analysis of physical behaviour, with accompanying mathematical models; block diagrams and sketches are used. Interesting to note is that present day computer tools do not use these product models. A reference model for the system and conceptual design phase is developed on the basis of the design process and the product models found. This reference model is a cube where the three main axes are simplex $\leftrightarrow$ composite; abstract $\leftrightarrow$ concrete; and mental $\leftrightarrow$ actual. Also, the distinction between formal and informal is made. This relates to how the transitions in the cube are made and is thus not an extra axis.

The second part contains the development of the FunKey architecting method. This method couples the *Fun*ctions and *Key* drivers using a coupling matrix $\mathcal{C}$. This provides for insight in the problem, a direct connection to TRIZ (the Theory of Inventive Problem Solving), and a means to track progress on different hierarchical levels of the design. The procedure is to make an inventory of the functions the system has to perform (in the rows of the matrix) and of the key drivers (in the columns). Then the coupling matrix $\mathcal{C}$ is filled. For each function contributing to a key driver, the corresponding cell is marked. Initially a cross ($\times$) can be used. When more information becomes available through expert interviews, additional modelling, reuse of information from previous experience, etc., the crosses can be replaced with numbers. Discrete numbers can be used, or particularly when the information available is inaccurate, symmetrical triangular fuzzy numbers (STFNs) can

be used. The STFNs have a triangular membership function around a central value. STFNs can be added, subtracted, multiplied and even RMS addition is possible. Several system architectures can be developed by allocating the functions to subsystems. Making several different architectures is recommended. With the numbers or STFNs a system budget can be derived for each key driver. In parallel to the creation of architectures and system budgets, the coupling matrix $\mathcal{C}$ is easily used to apply TRIZ. For this a connection with the TRIZ contradiction matrix is made. Additionally, the direct application of TRIZ principles is facilitated by introducing the positive and negative priority matrices. The TRIZ principles found can be applied to the system, the subsystem, and/or the function.

The third and last part describes how FunKey architecting is applied in three different ways. The first application is where an accurate positioning system for a wafer stepper has to be designed. In a participating manner, FunKey architecting is used to investigate the problem, to find new concepts, and to set up system budgets for overlay and throughput of a novel wafer stepper/scanner. The second application involves FunKey architecting for finding innovative concepts for waste baler systems. The use of TRIZ in combination with FunKey has provided many interesting concepts. Also an architecture is developed. The final application is by teams of Industrial Design Engineering students designing a computer controlled table-soccer game. The five teams have learned the basics of FunKey and were free to adopt the method or not. This way the performance of teams with and without the use of FunKey can be compared. The results of the three application cases show that FunKey architecting does meet the goal of the research project. In the wafer stepper and waste-baler cases, FunKey quickly led to insight into the problem and provided overview of the system. Also the innovation step is clear from the results obtained. In the student case, some of the teams used FunKey to create and maintain overview. These teams performed better than the ones not using FunKey, or not using it to its full extent.

In the last chapter it is concluded that the goal of achieving insight, creating and maintaining overview and stimulating innovation, has been met. Even more, the use of FunKey architecting to track the progress of a complex design project is advocated. As FunKey can be used to create decompositions on every level of the design hierarchy, it can also be used to analyse the impact that design results on a given level have on other levels. The earlier mentioned STFNs are very helpful in this respect. This way it is possible to sample the design process with little effort at a much higher rate than is achievable by reviewing documents. This is expected to reduce design risks.

Samenvatting

Producten die vandaag de dag op de markt komen bevatten elektronica, mechanica en software die nauw moeten samenwerken. Integratie is nodig tussen modules, maar ook binnen modules. Deze integratie is nog veel nauwer bij complexe systemen als verkeersvliegtuigen, wafer steppers en medische diagnosesystemen. Het ontwerpen van dergelijke systemen vergt een intensieve samenwerking tussen de elektrotechnische en werktuigbouwkundige ingenieurs en de software-ontwikkelaars. Door hun opleiding en ervaring gebruiken deze mensen verschillend jargon. Bovendien zijn organisaties tegenwoordig verspreid over de wereld. Hierdoor hebben de specialisten vaak slechts een beeld van hun eigen taak. Samen met de manier waarop computergereedschappen werken (van detail naar samenstelling naar geheel) levert dit te weinig informatie over de context van de taak. De systeemontwerper aan de andere kant, mist juist vaak detailinformatie die hij nodig heeft om de integriteit van het systeem te waarborgen. In dit proefschrift wordt een methode ontwikkeld die de detailontwerper van contextinformatie voorziet en die de systeemontwerper helpt bij het verkrijgen van overzicht en het toezicht houden op de detailontwerpen. Het doel is driedelig: *inzicht* verkrijgen, *overzicht* verkrijgen en behouden en *innovatie* stimuleren.

Het eerste deel van de dissertatie onderzoekt het ontwerpen van complexe systemen. Verschillende zienswijzen op het ontwerpen in het algemeen, en het conceptontwerpen in het bijzonder worden behandeld. Er is overeenstemming tussen de verschillende auteurs dat het proces is onder te verdelen in specificaties opstellen, conceptueel ontwerpen, embodiment ontwerpen en detail ontwerpen. De systeemarchitectuur, de ruggengraat van het ontwerp, wordt ontwikkeld in de conceptuele fase. Uit interviews met ontwerpers werkzaam in de conceptuele fase is een lijst met gebruikte productmodellen opgesteld. In de vroege fase van het ontwerpen worden systeembudgetten, analyses van fysisch gedrag met de bijbehorende wiskundige modellen, blokdiagrammen en schetsen gebruikt. In de computerprogramma's die voor de conceptuele fase ontwikkeld zijn, komen deze modellen niet of nauwelijks voor.

Vervolgens is op basis van de geïnventariseerde modellen en de kennis van het ontwerpproces een referentiemodel voor concept- en systeemontwerpen gemaakt. Dit model bestaat uit een kubus langs de assen simplex $\leftrightarrow$ samengesteld, abstract $\leftrightarrow$ concreet, en mentaal $\leftrightarrow$ reëel. Ook wordt het onderscheid tussen formeel en informeel gemaakt. Dit relateert aan hoe de kubus doorkruist wordt en is dus geen extra as.

In het tweede deel van de dissertatie wordt de FunKey architecting methode ontwikkeld. Deze methode koppelt *Fun*cties aan *Key* drivers met behulp van een koppelmatrix $\mathcal{C}$. Dit levert inzicht in het probleem, een directe verbinding met TRIZ (de Theorie van het Inventief Probleemoplossen) en een manier om de voortgang van het ontwerpproject op verschillende lagen van de hiërarchie te monitoren. De procedure die bij FunKey hoort is als volgt. De functies die het systeem moet volbrengen worden in kaart gebracht (in de rijen van de koppelmatrix $\mathcal{C}$), evenals de key drivers (in de kolommen). De cellen van de matrix wor-

den van een kruisje ($\times$) voorzien als er een bijdrage van de functie aan de key driver is. Als meer informatie over het systeem wordt verkregen, kunnen de kruisjes vervolgens vervangen worden door getallen. Daarvoor kunnen gewone getallen gebruikt worden maar ook, als de informatie minder exact is, symmetrische driehoekige fuzzy getallen (STFNs). Deze STFNs gebruiken een driehoekige lidmaatschapsfunctie rond een centrale waarde. Ze kunnen worden opgeteld, afgetrokken, vermenigvuldigd en ook RMS worden opgeteld.

Met de koppelmatrix kunnen verschillende systeemarchitecturen gemaakt worden, waarvoor met de ingevulde koppelmatrix ook systeembudgetten voor alle key drivers kunnen worden afgeleid. Ook kan op basis van de ingevulde koppelmatrix een verbinding met de contradictiematrix van TRIZ gemaakt worden. Tevens kan TRIZ direct worden toegepast met de positieve en negatieve prioriteitsmatrices. De op deze wijze gevonden TRIZ principes kunnen op het systeem, het subsysteem en/of de functie worden toegepast.

Het derde en laatste deel beschrijft de toepassing en evaluatie van FunKey architecting. Ten eerste is FunKey toegepast op de ontwikkeling van een nauwkeurig positioneersysteem voor een wafer stepper. Al meewerkend is FunKey gebruikt om het probleem te onderzoeken, nieuwe concepten te vinden en om systeembudgetten op te zetten voor *overlay* en *throughput*. Ten tweede is gepoogd om nieuwe concepten voor afvalpersen te ontwikkelen. Samen met TRIZ leverde FunKey verschillende interessante concepten op en is een architectuur ontwikkeld. Tenslotte hebben studenten Industrieel Ontwerpen FunKey kunnen toepassen in een ontwerpproject over een computergestuurd tafelvoetbalspel. Vijf teams hebben FunKey leren kennen en al dan niet kunnen toepassen op het project.

De resultaten van de drie toepassingen laten zien dat het doel behaald is. In het geval van de wafer stepper en de afvalpers heeft het gebruik van FunKey snel geleid tot inzicht en overzicht. Uit de gevonden concepten is duidelijk dat de innovatie ook gestimuleerd is. In het geval van de studenten verkregen de teams die FunKey gebruikten om inzicht te krijgen en overzicht te houden betere resultaten dan de teams die FunKey niet gebruikten.

Het laatste hoofdstuk bevat de conclusies en aanbevelingen. De belangrijkste conclusie is dat het doel van inzicht verkrijgen, overzicht verkrijgen en behouden en innovatie stimuleren gehaald is. Bovendien kan FunKey gebruikt worden om de voortgang van het systeemontwerpen in de gaten te houden. Omdat FunKey gebruikt kan worden voor het maken van decomposities op elk hiërarchisch niveau, kan het ook gebruikt worden om de gevolgen van beslissingen op één niveau op andere niveaus te onderzoeken. De eerder genoemde STFNs zijn hierin behulpzaam. Het ontwerpproces kan op een hogere frequentie bemonsterd worden dan mogelijk is met het reviewen van documenten. Op deze wijze wordt verwacht dat risico's verkleind kunnen worden.

Preface

Scientific research in a field where one of the best known books is called "The Art of ..." is a perilous undertaking. When the current status is best described as an art, there is apparently a lot of tacit knowledge waiting to be made explicit. This thesis aims at making a considerable contribution in that direction. Along the same lines there is a lot of tacit support given to this research. In this preface it is impossible to make all that support explicit. Nevertheless I give it a go.

This project has had better and worse times. The worse times were when the education and organisation of the Industrial Design Engineering curriculum started. Though I did not perceive these activities as "bad" at all, they required so much attention that only every now and then a little time was spent on research. As one can imagine this meant the project did not move forward. The better times started when I decided to commit myself to at least one day a week of research. From that moment on, progress was made and results were obtained. I thank my promoter, Fred van Houten, for stimulating me to really spend those days on research. The freedom you gave me to shape my research according to my interest has shown your trust in me.

The time spent with all the colleagues at the "OPM-koffietafel", be it drinking coffee, eating cake, just talking, or discussing education and research in general, and the situation in the world in particular, has been great. This was one of the reasons I have always liked going to work and I expect that this will continue.

The two years I could contribute to the wafer positioning system for MAPPER lithography has been a great time. The colleagues of the MAPPER wafer positioning team provided a very good atmosphere in the "aquarium", even though that room was also known as "the sauna". Also Guido, Halbe and Bert-Jan deserve to be mentioned here for the great technical and personal discussions. For the same reasons, I like to thank the people at Demcon. Although I came there only every now and then, it felt as being part of the crew. In particular Rini, Sjoerd, Peter and Denis have contributed to this. It is therefore great, Denis, that you are a member of my promotion committee.

The discussions with Gerrit Muller when we both worked at ASML have started my interest in thinking about the system design process on a higher level than just "doing the job". I am therefore very pleased that you are in the promotion committee. I thank you for the feedback and discussions on this research.

It is an honour to have Anton Basson in the promotion committee. Thanks to the connection between our research and interests, we have had interesting discussions during and after your sabbatical here in Twente. I appreciate your effort for coming very much.

The other members of the committee, Prof. Tetsuo Tomiyama from Delft University, Prof. André Dorée and Prof. Job van Amerongen, were less involved in the research. Yet your comments, feedback and time are much appreciated. I think there is enough mutual interest for future contact.

The Master student that contributed most directly to this research, Willem Alink, deserves to be mentioned here. The stimulating discussions and feedback both ways were great. I still think it a pity that I never had the chance to hear you sing... Also the students from the DMS project in 2007 are thanked for their work.

Maaike Nijkamp has lent her talent in creating the cover image of the thesis. It was a very vague question I asked: Can you make a "nice" drawing from two very schematic line drawings. You have taken it seriously and this has resulted in the artist impression of the FunKey method on the cover. I thank you for helping me in designing the cover, too.

Some parts of the thesis have been checked for the English by John Cornelius. John, you are very busy, therefore your time is much appreciated.

If it wasn't for my parents, this thesis would not have existed. You have raised me and taught me the most important things in life. And you also supported me in going to Australia for my practical training; to Delft for broadening my knowledge; to Eindhoven for my first job; to Switzerland for marrying Lilian; and finally back to Twente to step in my father's footsteps. It is therefore tragic that my father passed away so short before I could finish the promotion. I hope to honour my wise and loving father by dedicating this thesis to him; and with him I honour my mother.

Joris en Casper, wat zijn jullie toch twee lieve *boeven*. Het plezier en de energie die jullie uitstralen maken me een trotse vader. Het laatste jaar zijn veel dingen blijven liggen. Daar gaan we nu verandering in brengen.

Lilian, als ik jou niet had ...
Ik zeg het geregeld en dat maakt dat het zo gewoontjes klinkt. Maar het is echt zo. Als ik jou niet had, dan was er heel wat minder plezier in het leven en zou ik veel dingen niet kunnen of willen. Je zorgt voor een stabiele basis en stimuleert me om dingen te ondernemen. We hebben veel mooie dingen gedaan waarop we samen kunnen terugkijken. Laten we zo door gaan en nog veel mooie dingen beleven.

... thanks be to God, which giveth us the victory through our Lord Jesus Christ.
Bible, 1 Corinthians 15: 57
The Messiah by G.F. Händel

Hengelo, 25 February 2008 Maarten Bonnema

Contents

Chapter 1

Introduction

How can concept and system designers in present day development organisations be supported? That is the question addressed in this thesis. As an introduction, the fields of research and application are briefly discussed by looking at the problem at hand. System and concept designers deal with a large amount of diverse information. This information has to be handled, organised, processed and recorded. As the designer does not work alone, he or she has to communicate with colleagues. These colleagues being fellow system designers, and detail designers or specialists.
An important issue in this matter is the generation of the system architecture for a complex system under design. A definition for system architectures is derived.
We will describe the way others have approached this problem. Based on this the current research is described.

Designing complex products is getting more complicated. On the one hand because of the increased number of product functions and increased difficulty of these functions. On the other hand, it can be observed that the number of people involved has increased [Schulz and Fricke, 1999]. An additional aspect of this is that these people are increasingly more distributed over the planet. We will look at the characteristics of this problem, and will provide means to handle them. We will use the development of complex systems like medical imaging devices and wafer steppers as a target.

1.1 The Field

If present day organizations for development of high-tech equipment and products are regarded, we see that the products they generate are complex. Moreover, these products have increased in complexity over the years. Because of politics, national interest or historic reasons, the organizations that create these products have become more complex too: companies have merged, have outsourced parts of the development etc. This is in contrast with the way of working, implemented in tools and methods that are largely identical to years ago. Although a change has taken place with the introduction of computer aided design (CAD) in the mechanical design practice, this change has not worked for the better [Ottoson, 1998]. The author of that reference contends that in the "drawing board environment", the work flow was from "big picture" to detail. With the big picture, in the form of the large overview drawings, always present. In the days of CAD, the models are built up from detail to assemblies to modules to systems. The notion of the big picture is largely absent in the first detailing. In combination with the use of computer displays that are still small, but fortunately ever less so, the overview over the design has decreased: a computer display shows details very well, but the actual size and the big picture it has to fit in are hard to estimate from a screen. This holds for the mechanical discipline. However, as will be presented below, present day systems consist of mechanics, electronics and software that have to cooperate intensively. CAD systems that can work with such multidisciplinary products are just being researched [Lutters-Weustink *et al.*, 2007].

[Axelsson, 2002] contends that tools and methods do not design systems; humans do. Although there have been attempts at creating automated design systems, we will focus on ways to help the designer, but not take the innovative steps from the human responsibility. We feel that by helping the designer to acquire necessary knowledge about the ongoing design, and to order that information, he is able to make better informed decisions and use his creativity more efficiently, resulting in better products.

1.1.1 Real-life illustrations

If we look for example at the development of aircraft by Airbus, we see a total of nine design centres, of which seven are located in Europe. These design centres are closely related to production facilities that deliver modules to the two final assembly lines in Germany and France. Although there are differences between aircraft models, mostly the fuselage is made in Germany, the wings in the United Kingdom, the tail in Spain and the front and centre sections in France. As the aircraft are typically large-sized structures, special transportation means had to be designed. Two of them being the well-known "Super-Guppy" and "Beluga" aircraft (`www.airbus.com`).

As each of the design centres contains its own personnel, with different cultural backgrounds, the culture differs from one centre to the other. The centres are spread over Europe; therefore face to face discussions with the designers are hard. Also the use of tools and methods, or versions of the tools, may differ. Due to these differences it may be hard for system designers to keep overview, and to track changes that may have effect on the overall concept of the system under design. These issues have been causing serious delays in development and delivery of the A380 aircraft (see [van der Heide, 2007a, b; NRC, 2007]. The cause of the delays was not in the difficult items, but in by itself simple parts: the wiring [van der Heide, 2007b]. In a comparable system mentioned in the same reference, the Boeing 787 Dreamliner, bolts and nuts created problems. The difficulty has been the fact that these in themselves simple components are connected to many other systems.

Another example is Philips Medical Systems (PMS) in the Netherlands. This company mainly develops and manufactures medical imaging systems like CT and MRI scanners, and has several development centres around the world: Best (the Netherlands), Hamburg (Germany), Helsinki (Finland), Bangalore (India), and Cleveland (US). Partly because of history; PMS has acquired several companies in the same field of industry, partly because of cost reduction; software development in India is cheaper than in the Netherlands.

1.1.2 Observed trends in design of complex systems

In both cases the products to be designed are complex, business to business products (medical imaging systems and commercial aircraft). When we compare them to the state of the art products of two decades ago, there are striking differences:

- Function integration has increased, both on the product scale (products are able to perform more functions) as on the part scale (a part performs several functions);
- Many functions are performed automatically or even autonomously;
- Products contain mechanics, electronics and software that have to cooperate intensively;
- The time-to-market has decreased;

- The team of designers is large and may be at different locations around the world;
- Products are part of product families with similarities and differences, and reuse of (software) components;
- The expectations and requirements on quality, reliability and maintainability have increased.

Present day designers are confronted with these trends that complicate their work, also see [Calvano and John, 2004]. Therefore a shift of emphasis in the design process is needed. As known, the largest portion of the costs is made in manufacturing and preparation for manufacturing. However, the costs of the product are largely defined in the early phases of design: system design and conceptual design, see [Kals *et al.*, 1996, p.17 and Zuo and Director, 2000].

Due to the higher expectations and tighter requirements, the processes that have been used before in creating products will be enormously over-stressed because of these trends. In particular the way development teams used to be organized, in disciplines, hampers the development of products that require the synergistic cooperation of these same disciplines. Because of this the organization in many companies has been modified into multidisciplinary project teams ([INCOSE SEH Working Group, 2000; Zwikker, 2007] and personal experience). These teams consist of developers from mechanical, electrical and software engineering background. Depending on the product at hand, also ergonomics specialists, marketeers and other disciplines are part of the team. As can be seen in [INCOSE SEH Working Group, 2000], this way the development time and costs can be reduced significantly.

Communication between the disciplines, inside and among these teams is crucial. There are organizational measures that reduce the barriers, like collocation, project meetings etc. These measures do not eliminate the different jargons and do not create a synergistic way of working, though. There is a difference in thinking between for instance mechanical engineers and electrical engineers. As mentioned by Eising [2007] even different mechanical engineers have a different view on their object of interest. These different views, and the absence of a common view, result in different optimizations (and thus different designs).

As the detail information is in CAD systems, it is hard (or even impossible) for the system designer/engineer to obtain the information that will provide him with overview of the current state of the design. We will develop a simple tool that can be used for this.

1.1.3 Background

The Laboratory of Design, Production and Management of the Department of Engineering Technology at the University of Twente originates from production technology research. Over the past decades the focus has shifted from this technology oriented research (the 70's and 80's), via design engineering (90's and 00's) to research on application, usability, concept design and systems design (00's). Central in this shift has been the fact that design gets more multidisciplinary, as described above, and needs more focus on the ability to solve problems: moving from technology oriented to application oriented research. The thesis emerges from research on systems design and conceptual design. In this track, it has been researched how concept and system designers can be supported.

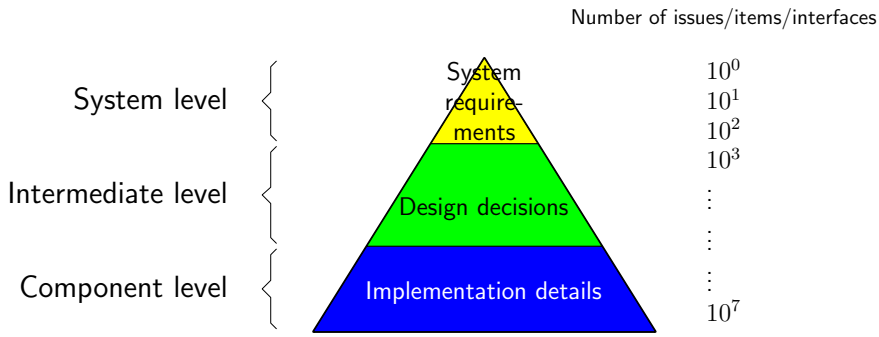


Figure 1.1: Pyramid showing the increase of issues when moving in the direction of detail design. After [Muller, 2007].

1.2 The Problem

In Figure 1.1 the problem is shown. At the top of the pyramid there are only a few requirements at a high level of abstraction. These stem from the stakeholders. The requirements describe their need. The complexity here is not too high. Moving down in the pyramid, those requirements spread through system requirements, subsystems, components, and finally to specifications and detailed design. At the bottom of the pyramid, the system is really complex: there is an enormous number of items to consider. Fortunately this is mostly mono-disciplinary (software, or mechanics, or electronics etc.). A large team of trained designers can handle this when each designer is given a problem that is, as much as possible, uncoupled from the rest of the system. Total decoupling is impossible as a system consists of cooperating parts. However, as is shown in [Suh, 1990] minimizing the coupling should be aimed at. If a totally decoupled design can be found, the system is no longer complex. It can even be argued that it is no longer a system, but merely a set of cooperating devices. As described in [Blanchard and Fabrycky, 1998, p.2] a system consists of elements that "cannot be divided into independent subsets." A consequence of this is that the behaviour of the system as a whole cannot be described by the separate behaviour of the subsystems or components (that is the reductionistic approach), but it *emerges* [Lewes, 1875] from the cooperation of and interaction between the subsystems or components.

At the system level, this is the top of the pyramid, the number of issues is relatively limited and can be dealt with by a few persons. However, how the decomposition from system level to component level is created is difficult. There are no tools or methods to achieve this. Because of this lack of tools and methods, most information runs one way: from top to bottom. The required accompanying stream of information that is needed for the system engineer to maintain overview, running from bottom to top, is most often lacking, or is created in an ad-hoc manner by system engineers that walk around the design department, probing the different parts of the design process.

The central issue of this thesis is how can designers from different disciplines work together effectively, how can the problem of designing a large and complex system be divided into smaller, more manageable parts, and how can the fit between these parts that

are designed by multidisciplinary teams be guaranteed. As we will see in Chapter 2, gain is expected if the conceptual phase of the design process is undertaken with the principles of systems engineering.

An important term here is the *system architecture*. For clarity, a short introduction to the term system architecture will be given first.

1.3 System Architecture

In environments where complex systems are designed and produced, architecting is an essential step in creating the systems [Gulatti and Eppinger, 1996; Maier and Rechtin, 2000; Muller, 2004b]. Complex systems are by definition systems that perform many functions. Designing moderately complex systems can be supervised by one person who ensures the proper fit and cooperation between the constituting parts. For present day highly complex systems this is impossible. A team is required because the only way to create these systems successfully is by *divide and rule*. The determination of the division-lines is what system architecting is about.

In [INCOSE SEH Working Group, 2000, p.10] a system architecture is defined as:

> "The arrangement of elements and subsystems and the allocation of functions to them to meet system requirements."

This definition describes the system architecture within a system. However, we feel that an architecture should expand beyond the system itself. For every function it should be considered whether it is performed by the system to be designed, the user, or the environment. In particular when using TRIZ [Altshuller, 1997, 1999] moving a function to a super system or to the environment, is to be considered. Where a super system is a system on a higher hierarchical level. This is relevant when the system to be designed is part of another system; designing is in most cases a hierarchical process. According to [Blanchard and Fabrycky, 1998; INCOSE SEH Working Group, 2000] the super system is part of the environment. We like to distinguish between these two, as the super system often is designed in parallel to the system under consideration. Thus changes to either the system (and its parts) and the super system are possible and should be considered. Changes to the environment, on the other hand, are in most cases impossible. Analogously, functions can be allocated to the user. In addition to the partitioning, the system's *performance* has to be divided over its constituting parts as well. Therefore we propose, in line with [Gulatti and Eppinger, 1996], the following definitions for an architecture and system architecting:

System architecture defines the parts constituting a system and allocates the system's functions and performance over its parts, its user, its super system and the environment in order to meet system requirements.

System architecting is the process of defining a system architecture.

Although it seems contradictory, system's functions *can* be allocated to either its super system or the environment. For instance a hard disk drive needs to be cooled. This function is not performed by the drive itself, but by its super system: the computer system it is part of that has a cooling subsystem. Of course, the requirements for the drive have to state its operational conditions to ensure proper operation.

Also note the definition of a function [Blanchard and Fabrycky, 1998, p.62]:

Function A specific or discrete action that is necessary to achieve a given objective.

Note that the impact of the architecture goes beyond the product. The organisation that creates the product has to adapt to the architecture [Gulatti and Eppinger, 1996].

As mentioned in [Muller, 2004b, pp.33 and 34] which observation is also confirmed by [Hurley, 2004], the current state of support for system architecting is minimal. The process is performed by people that have gained experience in architecting one way or another. Methods or tools for architecting are not used explicitly. Either because they are not available, or are not applicable for the problem at hand. As the complexity of products increase, the time to market decreases and the financial demands make failures less acceptable, there is an increasing pressure on designers. Support of the crucial system architecting and conceptual design phases may well reduce that pressure. Such support should help the designers in the difficult and/or tedious tasks and should stimulate communication between them. The support tool should also be used to inform the detail designers and provide them with relevant context information.

1.4 The Approach

The issue at hand has had the attention of other researchers and research groups. We will look at these briefly before presenting the approach we have undertaken.

1.4.1 Other Approaches

Depending on the place in the design process, several methods have been introduced to integrate different disciplines. For instance, on the physics level tools like 20–Sim [van Amerongen and Breedveld, 2003; Controllab Products, 2006] have been developed to model the mechanics, electronics, thermal etc. domains in one tool. These tools use the analogons between different domains.

Computer Aided Design (CAD) tools have been used to integrate mechanical parts designed by different designers and even different companies. Also integration with simulation tools like Finite Element Modelling (FEM) is available. This has shortened the mechanical design loop: the designer can now easily see the strength, stiffness and dynamical properties of his design. Adaptations can be applied and a new calculation can be made.

CAD tools nowadays use features to model the parts and assemblies [Salomons, 1995]. A related research field is the application of mechatronic features [Lutters-Weustink *et al.*, 2007]. Here the connection between mechanical features and mechatronics will be made to design functions.

Others use Artificial Intelligence (AI) to aid in the design process [Bracewell *et al.*, 1993; Gero, 1987; Taleb-Bendiab, 1993]. We feel that using the designer's intelligence as much as possible, and support him in difficult and/or tedious tasks is a better way because that maintains the interesting and creative part of the work, while taking away the hard work.

A common element in these approaches is their bottom-up nature. To apply any of these, it is required to already have a partitioning of the system under design and a clear view on the functions to be performed.

A way to create such a distribution is to carefully regard the functions of the system and then find solutions. A well-known technique is the morphological scheme. Creativity

techniques (lateral thinking, brainstorming, synectics) can be used to find solutions for each of the functions in the rows of the scheme. Combining the functions will produce a system that can fulfil all required functions. How these functions have to be divided over the system in the different subsystems is not supported.

[Tomiyama and Umeda, 1993] introduce Function-Behaviour-State (FBS) modelling. A distinction is made between function and behaviour. According to [Tomiyama and Umeda, 1993] function is defined as:

> "A description of behaviour abstracted by human through recognition of the behaviour in order to utilize the behaviour."

This definition differs significantly from the definition we use (see page 5) in that Tomiyama and Umeda define the function by looking through human eyes at behaviour. We try to define a function as objectively as possible. There is a close relation between Tomiyama's *behaviour* and our *function*. According to [Tomiyama and Umeda, 1993] different implementations for the system can be found using the FBS model. However, how the functions can be distributed over the subsystems and components of the system is not treated. This creation of subsystems is treated in [Maier and Rechtin, 2000]. As this book approaches systems architecting as an "art", it relies mainly on application of heuristics.

1.4.2 Our Approach

The approach presented in this thesis differs from the ones above in that we abstract from the concrete components as much as possible. By identifying functions, relations between functions and analysing the requirements, a partitioning of the total systems' functions is achieved. Requirements for the subsystems can be derived using system budgets. This creation of architectures is the central theme.

The philosophy has been to not create a fully automated (or even: autonomous) system that solves architectural problems, but a system that helps the designer to (see [Salter and Gann, 2003]):

- capture and organise his thoughts;
- create several architectures;
- evaluate them;
- present them to others to feed discussions.

We feel namely, that the human designer(s) has the best abilities to make the final decisions. The method and tool have to enable the designer to use all information relevant at each moment during the design process. As not all information is relevant during the entire process, the tool may hide or highlight information at times (also see [Lutters, 2001]). In other words: we aim at providing the designer tools to organize his thoughts, communicate about them, and get suggestions for improvement.

From the above we conclude that three aspects are important in this approach:

- Insight (both in the problem and the context);
- Overview;
- Innovation.

Insight relates to in-depth knowledge. Overview can be subdivided into "create overview" and "maintain overview". It means a broad, comprehensive view. However, we are confronted with the fact that humans can only handle a limited number of inputs; the magical

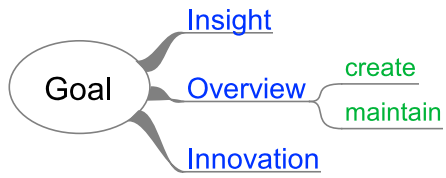


Figure 1.2: The goal of this research project.

number seven plus or minus two [Miller, 1956]. Despite the argument over the validity of this reference in the present field, we can conclude that there is a limit to the amount of information that can be easily handled by humans. Looking at Figure 1.1, with this limitation in mind, leads to the conclusion that it is impossible for one man to handle all detailed information in the design at hand. The problem needs to be subdivided in order to cut it down to pieces of the design process that can be handled by one or a few tightly cooperating persons. In [Axelsson, 2002] the fact that more than one person is required to design complex systems is even part of the definition of a complex system. We will investigate complexity further in Section 4.5.1. Innovation in this sense means to help create ideas that can lead to innovative products. We aim at developing a tool that can be used to achieve these goals, as shown in Figure 1.2.

1.5 The Thesis

Three main parts constitute the remainder of this thesis. In Part I, we will look at designing complex systems in order to define the context of this research and to investigate the aspects described above. For this, conceptual design and systems architecting will be investigated in Chapter 2. Based on literature several design *process* models will be treated. Chapter 3 investigates the *product* models used by designers. By interviewing designers about their work, a list of product models is derived. Using these models we will derive a reference model for concept and systems design in Chapter 4.

The second part treats the development of a method and accompanying tool for the early part of the concept and system design phase. This tool, FunKey architecting (from *Fun*ctions and *Key* drivers), is treated in Chapter 5. This tool, as the name suggests, bases on an inventory of the functions the system under design has to perform, and the key drivers that express the customer's interest. The connection between the functions and the key drivers is analysed using a coupling matrix. Using this matrix gives insight and overview and provides an easy start for creating system budgets. Chapter 6 extends the tool by connecting to the Theory of Inventive Problem Solving known as "TRIZ" [Altshuller, 1997]. This opens the path to innovation.

The third part shows the tool in practice and the evaluation of its use. FunKey has been applied (Chapter 7) in two industrial cases and one educational students' project. We will return to the aspects in Figure 1.2 to evaluate the research. This results in the benefits and drawbacks of the tool, and conclusions and recommendations that will be treated in the final Chapter 8.

I Designing Complex Systems

Chapter II

Conceptual Design and System Architecting

In literature both prescriptive and descriptive design process models have been proposed. This chapter will give an overview before concentrating on the main subject: Conceptual Design, a subject that has had less attention than the detail design phases. Particularly in high-tech environments specific conditions apply to conceptual design. This chapter will deal with these.
In high-tech environments, systems engineering is often applied. The chapter identifies the crucial role of the systems engineer in the conceptual design phase.
The design tasks and their results will be treated: specification; idea generation; evaluation and selection. As the conceptual designer is best aware of the overall functionality of the system and the interrelations, he is able to make a good decomposition into subsystems. Additionally, safeguarding the concepts is an essential task. This task is often neglected, leading to problems in the detail design phases or, even worse, in the prototyping and testing phase. In these latter phases, troubleshooting is needed. Often, the conceptual designer is the person with the best knowledge of the overall design. Therefore, his involvement in these phases is essential.
Finally, some remarks are made about documentation, an important aspect of any phase of the design process. Even more so for the conceptual design phase, as many important and far-reaching decisions are made by only a few persons. Some or all of these persons may leave the design process before the detail design or prototyping phase has started.

2.1 Introduction

To be competitive in a high-tech market, a company has to bring out advanced products in short time and with minimal effort. To achieve this, the design process has to be short, yet effective. One way to achieve this is to spend more time in the early phase of the process, the conceptual phase, to ensure a proper concept and feasibility.

This chapter will explore this *Conceptual Design Phase* in high-tech environments after first looking briefly at the overall design process. Aspects that will be treated are among others: the level of abstraction, goals, starting points and the people and skills involved. Also, attention will be paid to the links between systems engineering and conceptual design. The information is based both on literature and practical experience of the author.

2.2 Design Process

The Design Process has been researched by many scientists. Several of the well known authors on this subject are Pahl and Beitz [1996], French [1985], Horvath [2000] and van den Kroonenberg [1992]. In [Pahl and Beitz, 1996, pp.10–25] and [Lutters, 2001, §2.4] several of the design process models are described and compared.

This chapter is a reworked version of the paper published as [Bonnema and van Houten, 2003] and [Bonnema and van Houten, 2004].

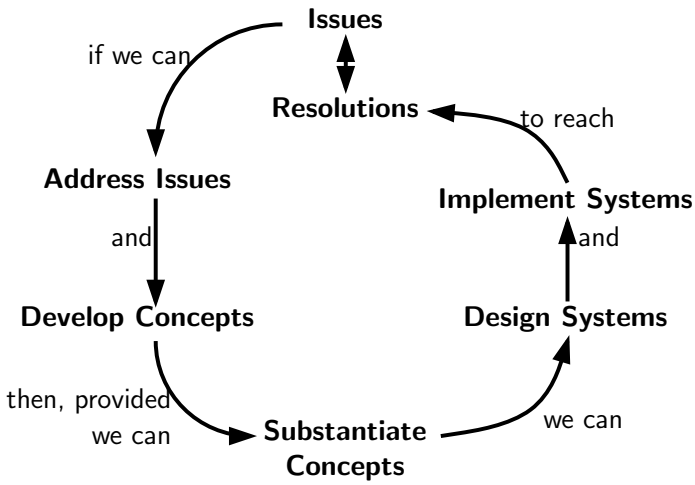


Figure 2.1: The Systems Design Process according to [Hitchins, 1992, p.xiii].

2.2.1 Conditions and Steps in Designing

Before looking at one of those models, it is wise to discuss the essence of design. Figure 2.1 shows one view on design: The goal of design is to find *resolutions* to *issues*. The right side of the loop shows the steps to be taken before resolutions can be brought to existence. On the left side conditions to these steps are shown. Please note that the first condition is that issues can be addressed. In other words: an issue has to be recognised as one. In many cases this requires a mind-set not many people have by nature. On the other hand, some people have the right mindset, but fail further in the loop; their creations can be qualified as *Chindogu* [Kawakami, 1995].

The second step involves the development of *concepts*; that what conceptual design is all about. Note that this is mentioned as a condition, not as a process step. So the designer has to be enabled to develop concepts. This is an easy thing to say but it raises the following questions as to how to develop concepts:

- What is a concept?
- What characterises a good concept?
- How is a concept generated, or: What are the steps conceptual designers take to come to a proper concept?
- How many concepts are needed before choosing one?

In Sections 2.3, 2.4 and 2.5 these questions will be addressed.

The developed concepts have to be substantiated* in order to be of use. This means a concept is only valid when its feasibility has been shown. Problem is that feasibility is hard to guarantee early in the design process. This will also be addressed in Sections 2.3–2.5.

*sub·stan·ti·ate: (1) To support with proof or evidence; verify: substantiate an accusation. (2) To give material form to; embody; To make firm or solid. (3) To give substance to; make real or actual. (The American Heritages Dictionary of the English Language, Fourth Edition)

When the conditions mentioned above are met, the design can be detailed, made and implemented. With a proper design, the issue under consideration may have been resolved. This of course, has to be verified and tested. When the results thereof are positive the issue has been successfully resolved.

2.2.2 Level of Abstraction

To be able to focus on the conceptual phase, it is important to explore the relationship of the designer with his problem. In [Alexander, 1966, pp. 75–78] three situations are described (see Figure 2.2).

Design in an *unselfconscious situation* acts directly on the artefact to be designed (see Figure 2.2a). The designer is the same as the workman that makes the artefact and the same as the end-user. To use the designations of [Alexander, 1966]; the fit between *context* and *form* is optimal because any misfit is detected by the user, resolved by the designer and implemented by the builder, because these three roles are performed by one person.

In a *self-conscious situation* the designer is a specialist in the field of designing a specific class of artefacts (e.g. architects design houses, mechanical engineers design bridges, electronic engineers design amplifiers). However, he has only limited experience in producing and using these artefacts. This means he first develops a conceptual picture of the environment and designs a conceptual picture of a solution (see Figure 2.2b). This conceptual solution is then translated into a form (a product[†]). This normally is done by someone else than the designer or the user. The fit will be worse than in the former situation due to the translations and due to the larger distance between the designer and the context of the product.

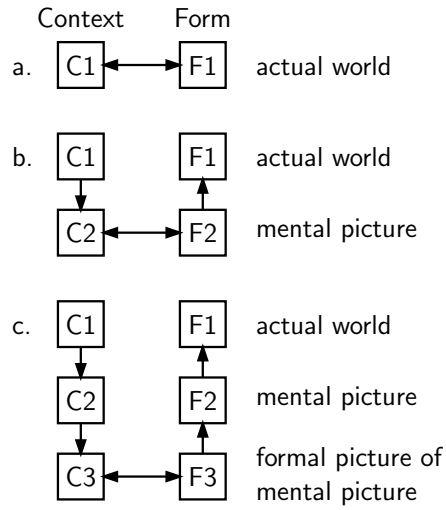


Figure 2.2: Three design situations according to [Alexander, 1966].

Is the problem more complex, which is generally the case in high-tech environments, then the designer will not only form a conceptual picture of the context, but he also makes a formal model of this conceptual picture (see Figure 2.2c). Based on this model, a model for the form is designed. This model is translated to a conceptual picture that will be implemented in the real world. Again, the translations from mental picture of the form to the conceptual picture, and from conceptual picture to artefact, are normally done by different people.

As can be seen, the distance between reality and the level of abstraction at which a solution is sought, and the number of translations increase with increasing complexity of the problem. According to [Alexander, 1966], this results in worse fit of the artefact to the context (also see [Sage and Armstrong jr., 2000]). In case of large-scale and complex systems the perceived distance will increase even further.

The solution that is proposed in [Alexander, 1966] is to explore the extensive and as complete as possible list of requirements in a systematic manner. The requirements are

[†]It should be noted that a *product* can be either physical or non-physical; e.g. a thing or a service.

grouped in a tree-like structure based on the effect each requirement has on a (group of) variable(s). It is believed that by arranging the requirements this way, the designer is easily directed to a solution. However, how such a solution has to be sought is left untreated. In present day complex systems the requirements space is so big that waiting before the requirements are fixed and organised will lead to a serious lack compared to the competition. While exploring the requirements, the systems has to be developed concurrently.

In [Schön and Bennet, 1996] this problem is addressed as well :

> "A designer makes things. Often, the thing initially is a representation, a plan, a program, or an image to be constructed by other people. Many of the relevant variables cannot be represented in a model; this limitation makes the design process inherently complex. A system is complex in the specific sense that, whenever I make a move, I get results that are not just the ones that I intend. That is, I cannot make a move that has only the consequences that I intend. Any move has side effects.
>
> This unpredictability is a central attribute of design — it is not necessarily the defining one, but it is important. It means that there is no direct path between the designer's intention and the outcome."

The view presented in Figure 2.2 shows the gap many designers face between conceptual idea and realization. It would therefore be an improvement when the behaviour of a possible (or prototypical) realization could be shown from the information in a concept. The perceived distance between the formal picture and the actual world would be reduced.

Again [Schön and Bennet, 1996] talk about this as well. While working on a problem, the designer will continually find new paths. When solving parts of the problem, new consequences, opportunities and side effects show up. These have to be considered. This would be made easier when the behaviour of the concept or system design could be studied early in the process.

2.2.3 Model of Design

Many of the design theories try to solve this problem, either by *describing* or by *prescribing* the process. Let us concentrate on a model of the design process proposed in a classic book on conceptual design [French, 1985], see Figure 2.3. Here, a *scheme* can be understood as a concept.

Although the steps are shown as distinct blocks, in reality they overlap. When analysing the problem, the need will become clearer; when making embodiments of the schemes, new schemes may be found, or the existing ones must be adapted. These overlaps and two-way connections are modelled by the feedback in the model. This is important to note, in particular in concurrent engineering; a strategy often used in high-tech environments.

The *Need* in this model is to be compared with the *Addressed Issues* in the model by Hitchins (Figure 2.1). It shows there is a situation that will be improved when a new product would exist. Establishing the need and describing it is often done by marketeers. In Figure 2.3 the need is taken as the input to the design process. Since we will concentrate on conceptual and system design, we will take the need as an input as well.

Of course it should be noted that the description of the need as indicated by marketeers will, in many cases, be insufficient for designers. Elaboration and further study is needed by

the designers in order to be able to design the needed product. This is indicated by the block *Analysis of Problem* in Figure 2.3. The result of this activity is a *Statement of Problem*. Many designers will call this the *requirements* or the *specification* of the object to be designed. It states, in the designer's language, the starting points, border conditions, performance figures etc. of the needed object.

The next blocks, *Conceptual Design* and *Selected Schemes* will be treated in Section 2.3 and will therefore not be elaborated upon here but it is worthwhile to give a citation from [French, 1985, p.3] about the conceptual design phase:

> "It is the phase where engineering science, practical knowledge, production methods, and commercial aspects need to be brought together, and where the most important decisions are taken."

When a concept (or concepts) has been selected it has to be developed further. French calls this the *Embodiment* and *Detailing* phases. The result of these is a set of drawings, a CAD-model, a production plan, and facilities layout etc.

The phasing shown in this model is also often present in other models of the design process:

- Specification and Requirement development;
- Conceptual Design;
- Embodiment; and
- Detail Design.

The model by French will be used here as a reference for the design process. For more models and for a comparison between them, the reader is referred to the earlier mentioned references [Lutters, 2001; Pahl and Beitz, 1996].

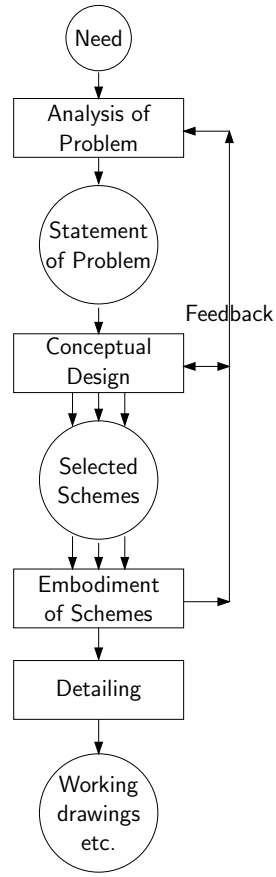


Figure 2.3: Design Process according to [French, 1985]. Rectangles indicate steps, circles indicate results.

In [Muller, 2004b] an interesting view on the creation of an *architecture* of a system is shown: The *Customer Objectives, Application, Functional, Conceptual, Realization* model, short: *CAFCR-model*, see Figure 2.4. The model can be used to describe the views on a system in relation to its context. An architecture is the framework wherein the designer has to find the concepts, see Section 1.3. So, an architecture can span several concepts, a concept can span several product instantiations. The CAFCR model is therefore a step in the direction of solving the problem shown in Figure 2.2 and side-input to the process shown in Figure 2.3.

In Figure 2.2c the route from context to form runs via mental and formal pictures. The content of these pictures is shown in Figure 2.4: the two left-hand blocks depict the mental and formal pictures of the context: *what* is the customer's problem and *how* does the customer want it to be solved? The two right-hand blocks describe the formal (conceptual) and mental (realization) picture of the solution. The functional block describes the what? of the product in order to solve the problems in a way that satisfies the customer.

By designing the architecture of the system based on the content of these blocks, the fit between form and context may be improved. This requires that the architect/designer hops frequently between the different viewpoints [Muller, 2004b].

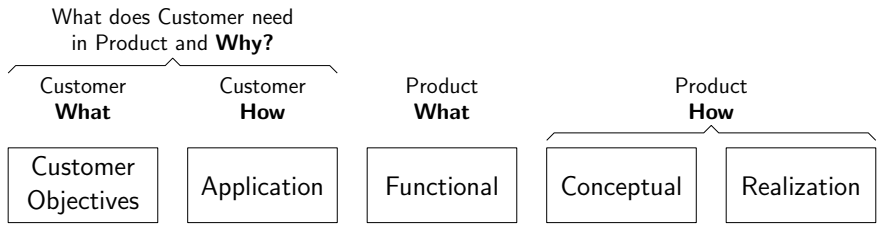


Figure 2.4: Basic CAFCR Model as presented in [Muller, 2004b].

In Table 2.1 the emphasis on the architectural aspects of the CAFCR-model in each of the phases of the design process is shown. Please note the diagonal shape of the table. Also, the conceptual phase is the only phase that covers all aspects of the CAFCR-model. In Chapter 3 this information will be used to determine what type of models can be used in the conceptual design phase.

2.3 Conceptual Design

WHY CONCEPTUAL DESIGN? — Designing complex systems, e.g. wafer steppers and aircraft, requires a large amount of effort, resources and time. It is essential to direct and concentrate the effort in one direction as soon as possible, but not *too* soon. One can compare this to undertaking an expedition or long journey. Before taking off, the chance of success has to be estimated, the area must be mapped and a proper route must be chosen. Based on this route and the expected obstacles on it, the right people can be hired (and trained if necessary), the materials and tools can be ordered and the expedition can start. Conceptual design can be seen as this preparation phase before the actual journey starts. The architecture of the journey can consist of the global division into day-trips with the approximate places to sleep and the principal means of transport.

The goal of the conceptual design phase is to explore the problem and the field of solutions, find the best solution that is feasible, work it out to a stage that the whole team can start working on it, plan how this solution can be implemented, and hire and train the people needed. All this has to be done based on the information presented in the

Table 2.1: Architectural aspects in design in different design phases. The number of +-es shows the importance of the aspect in the corresponding design phase. (CO: Customer Objectives; A: Application; F: Functional; C: Conceptual; R: Realization. Also see Figure 2.3 and Figure 2.4.)

	CO	A	F	C	R
Need	++	+			
Analysis of Problem	++	++	+		
Conceptual Design	+	++	++	++	+
Embodiment		+	+	+	++
Detailing					++

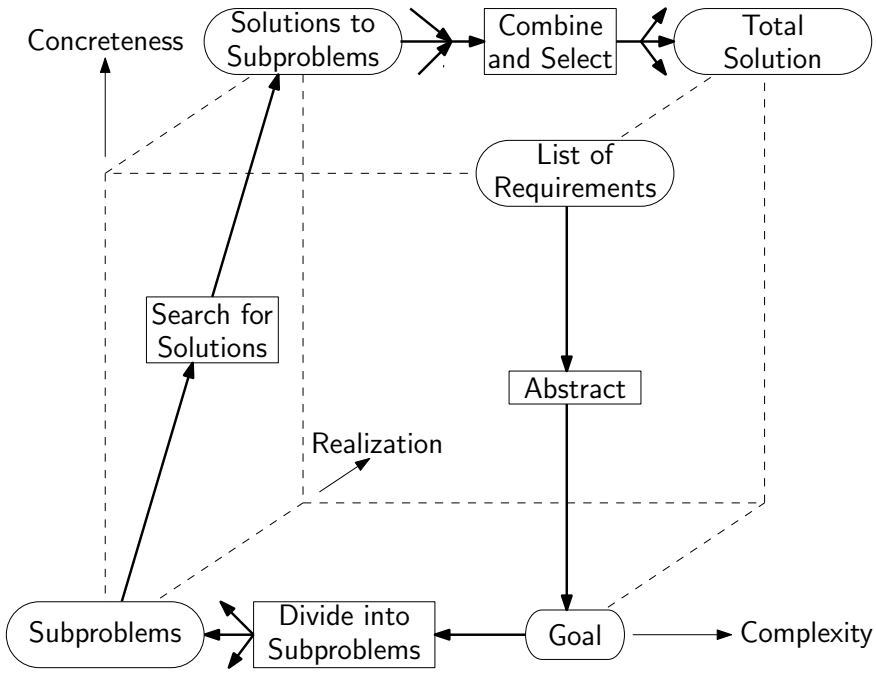


Figure 2.5: The space of the conceptual design phase [Krumhauer, 1974]. Ovals indicate information, rectangles show information processing.

need, scientific knowledge, experience, critical reasoning and discussions with people from marketing, production etc.

As can be seen in Figure 2.3, conceptual design takes place early in the design process where only the need and the problem are stated. In many cases the need and problem will become clearer while designing concepts. That is why every conceptual designer must incorporate reviewing the need and requirements in his task schedule.

On the other side of the conceptual design phase in Figure 2.3, one can see the embodiment phase. As the conceptual designer is best aware of the essentials, drawbacks and characteristics of the chosen concept, he is the best person to guard the concept during the embodiment and detailing phases. One small decision in the embodiment or detailing phase can completely ruin a concept. Therefore, the conceptual designer has to monitor the design process that takes place after a concept has been chosen. Depending on the complexity of the design, and the number of people working on it, monitoring can be done by one person or by a team.

In [Krumhauer, 1974] the conceptual phase is ordered on three orthogonal axes (the terms in brackets show steps along the axis):

- Concreteness (Physical principle → Working principle → Construction element);
- Complexity (Unit → Chain → Structure);
- Realization (Idea → Product).

The route of the conceptual design phase through the space created by these three axes according to [Krumhauer, 1974] is shown in Figure 2.5. Note that the starting point (List of Requirements) and the end point (Total Solution) are concrete and complex. The problem that is complex and concrete, but is not yet realised is dealt with by first abstracting (reduce concreteness) and then splitting into subproblems (reduce complexity). These subproblems can then be solved, thus increasing the realization and concreteness. By assembling the subproblems into a complete system, the complexity is increased. Where the start of the design process is concrete and complex but not yet realised, the solution is complex, concrete and realised. Important instruments are thus abstraction and splitting into subproblems.

Abstraction is also mentioned in [Kao and Archer, 1997] where several designers have been observed during a not too complicated design case. Three types of abstraction are identified: *horizontal*, *vertical* and *general*. Horizontal relates to complexity in the model in Figure 2.5, vertical to concreteness, whereas general abstraction is not directly linked to the model by Krumhauer. It is described as a combination of horizontal and vertical abstraction to ensure cohesion between the partial solutions.

In [Kao and Archer, 1997] it was noticed that domain experts design by reducing concreteness first, followed by reducing complexity. This conforms to the model in Figure 2.5. Also, the experts tended to work at a higher abstraction than non-experts. Non-experts did not use abstraction to its full extent. It was concluded that effective use of abstraction enhances design quality. Thus support of abstraction may help the non-expert in moving toward the results of experts. However, another possible conclusion from this study is that experts perform better in designing than non-experts, whether they are provided with support tools or not.

These observations are compatible with the situation shown in [Ottoson, 1998]. There it is mentioned that in the "drawing board environment" totality ruled over detail during design work. When the chief design engineer finished the systems design (as we would call it now), the senior designers focussed on subsystems, and later on the designers worked out the details. This describes reduction of complexity in Figure 2.5 and horizontal abstraction in [Kao and Archer, 1997].

RESULTS OF CONCEPTUAL DESIGN — What is a concept, then? In [Pahl and Beitz, 1996] no definition is given, but it is described as a *principle solution* or *specification of principle*. From the detailed description of the conceptual design phase it is clear that essential elements are working principles, a basic structure and calculations. [Pahl and Beitz, 1996, p. 68] does state that:

> "A lasting and successful solution is more likely to spring from the choice of the most appropriate principles than from exaggerated concentration on technical details."

[French, 1985] does not use the word *concept* but uses the word *scheme* instead. The terms in italics in the next quotation describe the characteristics of a good concept (italicisation by the author):

> "By a scheme is meant an *outline solution* to a design problem, carried to the point where the *means* of performing each *major function* has been fixed, as have the *spatial and structural relationships of the principal components*. A scheme should

> be sufficiently worked out in detail for it to be possible to supply *approximate costs*, *weights*, and *overall dimensions*, and the *feasibility* should have been assured as far as circumstances allow."

From this quotation it is clear that *functions* play a very important role in conceptual design. It is paramount to identify the functions the product has to fulfil, and to find relations between them. A principle solution for each function has to be sought. The configuration of these principle solutions then provide the basic structure of the concept.

[Krumhauer, 1974] distinguishes physical and operating (or: working) principles. A physical principle is the basic physical effect that is used (e.g. friction, lever, constant temperature during phase changes). A working principle is a physical principle plus a form in which it can be used (friction wheel, cogwheel, etc.). A concrete object that uses this working principle is called a construction element.

We will therefore use the following definitions:

Concept A structured set of working principles (to achieve a given goal).

Working principle or **Operating principle** A physical principle with a possible instantiation. The instantiation can be known or unknown.

Thus, creating concepts requires selecting working principles and structuring them. During the embodiment phase the working principle has to be detailed into a construction element, or a construction element that uses the chosen working principle has to be selected.

Having said this, do we need one, ten or one hundred concepts before proceeding? One hundred makes it very hard to rate them relative to each other. Also, developing hundred concepts to enough detail, knowing that 99 will not survive is a waste of effort. With only one concept, there is nothing to choose. Thus the minimum number of concepts is two. Several ideas may use the same concept (see the definition above), and therefore a continuum of ideas can be brought to a discrete number of concepts. Four to six concepts can be overseen, they can be made sufficiently different and they can be worked out far enough without wasting too much effort.

When the principles chosen are known from literature but new to the organisation a feasibility study is necessary for each concept. When they are entirely new, or known but applied in a new context, the feasibility study must be thorough. In many cases barriers in for instance technology have to be negotiated in limited time.

It is well known that early in the design process not many costs are made, but the largest part of the costs to actually build the system are defined, see [Kals *et al.*, 1996, p.17 and Zuo and Director, 2000]. Therefore it is important to make the basic design decisions very carefully. These decisions have to be made early in the process as to direct the design effort in one direction, focusing the effort of all designers; thus accelerating the development. It is therefore important to have a common basis for all designers to build and to elaborate upon (the baseline in systems engineering). In other words: one concept has to be chosen as early as possible, but not sooner than when the consequences of the decision can be overseen. Determining this point in time is a research issue in itself. Unfortunately reasons to revisit the concept choice may arise later in the design process. Although at such a moment one or more alternative concepts rejected earlier may look appealing, it should be noted that no effort has been put in finding flaws in these alternatives. Nevertheless, it can occur that reconsidering the concept choice leads to a better final design.

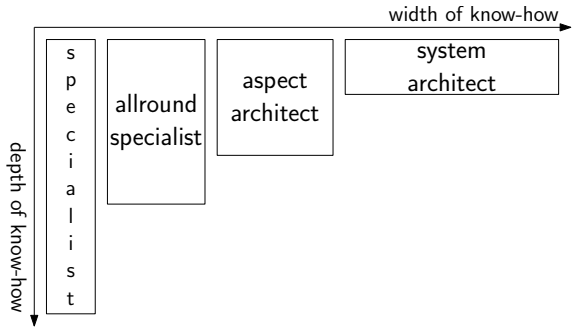


Figure 2.6: The width and depth of knowledge of different designers. The conceptual designer needs a wide knowledge with medium depth. Adapted from [Muller, 2005a].

2.4 People and Skills Involved

Although in some cases the conceptual design can be made by one person, in high-tech environments it is usually done by a team; small compared to the team needed to develop the concept into a detailed product, yet consisting of highly skilled people with different viewpoints. This section will take a look at these people, their skills and roles. Please note that several roles described below can be combined in one person.

From Figure 2.3 it can be seen that the conceptual design phase is based on the information in a statement of problem. This statement is created through analysis of the need. The Marketing department (or: *marketeers*) in the company are usually the ones stating that need. They should therefore be involved in the analysis of the problem. This ensures the problem as analysed by the designers is the one the marketeers diagnosed. Moreover, the marketeers may create the right awareness in the designers.

However, it is important that the analysis of the problem is taken to a wide enough extent so that the core problem is addressed. For instance, a need for a new *bicycle* directs the thoughts of the designers directly to a device we all know today, while in fact the need may be a new *environmentally friendly transportation system* (see Sections 5.5.4 and 6.3.2). This can have an entirely different appearance. Also the interfaces with the user and the (sub)functions may be entirely different. In some cases the need may even be resolved without a product at all. Though economical not interesting for the company, this is in many cases the best solution, of course.

When the problem is clearly defined, the essence is agreed upon by marketing and the design department, technical designers have to find the best suiting concept to solve it. As stated above, this involves defining the functions and subsystems of the product, arranging the functions over the subsystems and finding working principles that can be materialised. Figure 2.6 shows the different designers and their knowledge fields. All types of designers shown in Figure 2.6 may be needed in the conceptual design phase, although the ones with the wider view will play the major role as the functions and their relationships are the most important issues. The specialists involved will be needed for the most critical subsystems and when new technology is needed.

In the conceptual phase, the basic structure of the product is created. But not only that,

also the facilities and the process are to be considered in view of the total life cycle. *Systems Engineering* is vital in this. According to [Blanchard and Fabrycky, 1998] the following aspects are to be emphasised:

1. Improving methods for defining system requirements;
2. Addressing the total system from a life cycle perspective;
3. Considering the overall system hierarchy and interactions;
4. Organising and integrating the necessary disciplines; and
5. Establishing a disciplined approach with reviews, evaluation and feedback.

These aspects are particularly important with complex products, like the ones created in high-tech environments. According to [Sage and Armstrong jr., 2000] the systems engineering team acts as an interface team that provides the conceptual design and the technical direction.

From the description of the people involved, it can be concluded that the following skills are necessary (but not equally) in the team of conceptual designers:

- ability to communicate over disciplines;
- being eager to learn;
- having a clear view on the customer;
- ability to document;
- having a wide perspective;
- having an open mind.

2.5 Tasks

In Figure 2.7 the steps of the conceptual phase according to [Pahl and Beitz, 1996] are shown. Based on the specification, the problem is abstracted in order to find the basic functions required. If needed these are split into subfunctions. Solutions, in terms of working principles, are sought to fulfil the functions found. These are combined into useful systems. The alternatives are rated using criteria, based on the specifications, and one concept is chosen.

This process is fairly detailed. Each step shown in Figure 2.7 is detailed in the accompanying text. One can conclude again that in the first steps of the conceptual phase, functions play the most important role. These can be subdivided into subfunctions if necessary; where subfunctions perform a part of the main function. It may be needed to introduce *auxiliary* functions. These perform actions not essential for the main function, but necessary for practical reasons (e.g. removal of waste material). On the other side it is wise to distinguish *supplementary* functions; these perform other functions than the main function and can be seen as extra (see [Eekels and Poelman, 1995; Eger *et al.*, 2004a]). Later on the focus shifts to finding suitable working principles, and in the end to rating the found concepts.

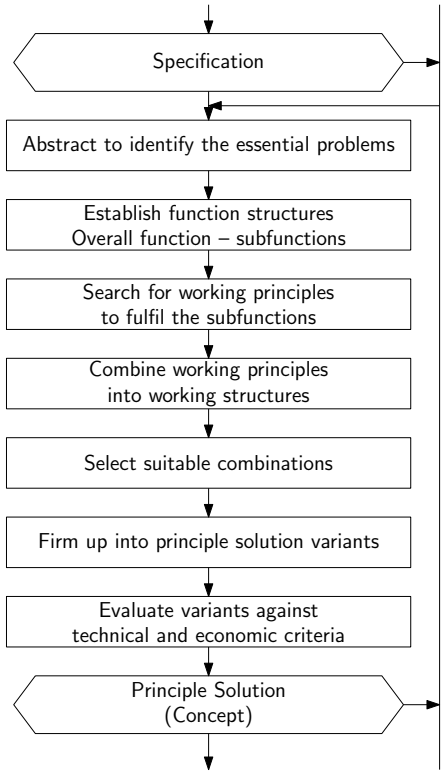


Figure 2.7: The Conceptual Design Phase according to [Pahl and Beitz, 1996].

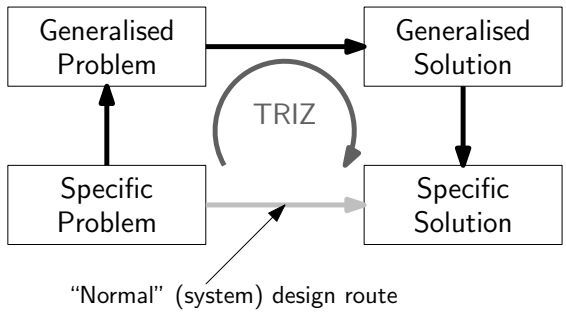


Figure 2.8: The approach of the Theory of Inventive Problem Solving (TRIZ).

In [Pahl and Beitz, 1996] the advice for the conceptual designer is to abstract as much as possible. When the requirements are listed, the task is to find the essence of the problem. This can be done by omitting requirements on extras, stating the requirements as qualitative as possible and then formulating the problem as neutral as possible. Please note that the list of requirements that this procedure is applied to has to be quantitative, exact and complete. The problem stated this way can be called the *goal* of the design procedure (also see Figure 2.5); it contains the essential information, without many distracting details.

In this context it is appropriate to mention the "Theory of Inventive Problem Solving", known under the Russian acronym TRIZ [Altshuller, 1997, 1999]. This theory that is invented by the Russian engineer Genrich Altshuller states, among others, that technical problems can be resolved using a limited set of patterns. These patterns have been derived from careful investigation of thousands of (Russian) patents. The general approach is shown in Figure 2.8. The problem at hand is translated into a generalised problem that is classified either by reformulating it into a *contradiction*, modelling it in a *substance-field model*, rating it on the *scales of evolution* or modelling it in the *9-window diagram*, among others. For each model several rules exist that provide a principle for solving the generalised problem. The generalised solution must then be translated into a specific solution for the problem at hand.

One of the well-known set of rules is the *contradiction matrix*. This uses the 39 generalised parameters of a technical system, inventoried by Altshuller (see Table 6.1 on page 73). Once a problem is reformulated as a contradiction between two of the 39 parameters, the improving parameter is looked up in the rows of the matrix, the worsening parameter in the columns. The indexed cell suggests then one to four *inventive principles*. Table 6.2 (p.73) shows a part of the contradiction matrix. The suggested principles (segmentation, local quality, prior action, etc.) are stated with a few simple words and are illustrated with examples, see Appendix C.4 and [Altshuller, 1997]. In most cases they trigger the designer's creativity into a promising direction. That is the distinguishing difference between TRIZ and most other creativity methods. TRIZ provides focus and direction for solving the current problem.

The TRIZ model represented in Figure 2.8 is also referred to as the "four-box" scheme. In [Nakagawa, 2006] an alternative description is given: the "six-box" scheme. This scheme does not put the steps to undertake in boxes, but models the data in boxes. The transforma-

tions are shown as arrows. According to Nakagawa this allows for a more uniform and less partial process. There is therefore an analogy to the CAFCR model (Figure 2.4) that represents the type of information instead of the steps to be performed.

Although this approach looks like a detour and appears to reduce creativity to application of rules, it is very powerful. Many examples exist where TRIZ has solved very difficult problems in an elegant and efficient way.

Returning to the process shown in Figure 2.7, two tasks are omitted that need to be addressed in the conceptual phase, both having a tight relation with systems engineering:

- Division into subsystems and/or -modules;
- Documentation.

The possibility to divide a concept into subsystems or -modules can be a key factor in determining which concept must be chosen. Although this issue is closely related to determining the functions, it needs explicit attention as, on the one hand, some functions can only be performed properly by a system-wide approach while, on the other hand, careful decisions on where to put an interface may aid in creating a properly working architecture.

To aid in the process of guarding the concept later on, important decisions have to be documented. Often when a design is worked out, problems occur that have to be solved. The solution must not conflict with the original concept. People like the conceptual designers and/or systems engineers can prevent such a conflict, but for the less complicated matters good documentation may suffice. Though documentation is not shown in Figure 2.7, the text of [Pahl and Beitz, 1996] does emphasize its importance and templates for appropriate documents are shown throughout the book.

As an example, one may look at a wafer stepper or scanner. Here *overlay* is one of the key performance-indicators. It describes the accuracy between the lateral position of two layers of an integrated circuit. Accurate overlay can only be achieved by carefully selecting working principles, and checking every decision against its possible influence on overlay. Note that overlay does not map onto one or a few specific functions of the wafer stepper; it is a characteristic of the system as a whole. In Figure 2.9 an inventory of the mechanisms that contribute to overlay are shown (also see [Muller, 2004c]). That diagram was made by the author in order to create an *overlay-budget* on the one hand, and to raise awareness among the designers about the importance of designing with overlay in mind on the other hand.

After the conceptual phase, it is important to monitor the embodiment and detailing phases in order to maintain the concept. It can be said, from personal experience, that one small decision taken after the concept has been chosen can ruin the proper operation. In particular when complex systems are to be designed it is important that one person or a small team keeps the system-wide view. When problems occur in the later phases, it is important that one of the conceptual designers is involved as a systems-engineer when solving the problem because of the possible danger of wrecking the original concept.

2.6 Conclusions

From the exploration above we can conclude that the results of the conceptual design phase determine the course of the design process to a very large extent. Based on descriptions of a need, the conceptual designers have to find a concept of a solution. First step is to define a goal. Then an architecture may be needed. When the architecture was defined before, it may

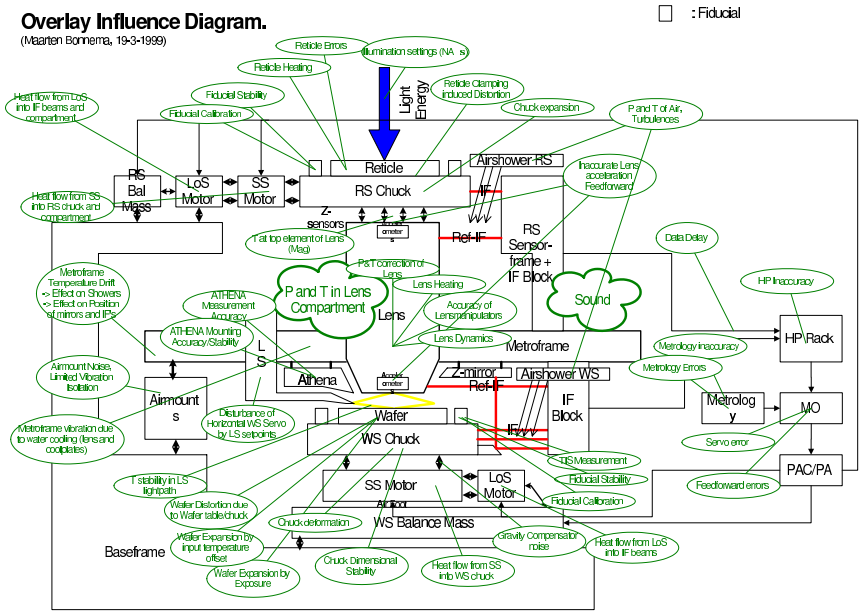


Figure 2.9: Concept of a wafer scanner: items influencing overlay are shown as clouds and ellipses in a schematic representation of a wafer scanner.

be reused, however, it must be reconsidered as the starting points for the architecture may have altered. The architecture serves as a basis for the concepts.

From a multitude of ideas, several *different* concepts can be derived. This number has to be reduced by checking feasibility so that the number of concepts before choosing one to develop further is some four (± 2). The number four is based on the fact that ten or more seems a waste of effort as nearly all work done will be destroyed by choosing only one out of the many concepts. When there is only one concept there is nothing to choose and compare. So depending on the maturity of the field, two to six concepts are required.

The moment when the choice for one of the concepts has to be made must be determined carefully. More research is needed to give guidelines for this. It is paramount that after a choice has been made everyone is determined to make this concept work. If not, manpower is wasted. However, leaning back every now and then to evaluate the choice can be very informative and instructive for future projects.

In the conceptual phase there are many activities. A support system for this phase must be helpful in those aspects that are difficult or require a lot of work. It must be investigated what these aspects are.

In the next chapter, as complement to the process models treated in this chapter, the use of product models by designers in the early phase of the process will be investigated in order to get a better view on how these designers work.

Chapter III

Use of Models in Conceptual Design

The chapter investigates the use of product models by conceptual designers. After a short introduction, abstraction applied in conceptual design is described. Applications of conceptual design from literature are used to identify several product models used by conceptual designers to handle the complex problems. Next, the models available in four conceptual design support tools are listed.
To investigate the current use of models by conceptual designers a questionnaire has been designed and issued. The design and results of this questionnaire are described and analysed. The results have been discussed with conceptual designers. It is concluded that several types of models are needed for conceptual designers to cope with and structure the large amount of information. In particular budgets are used in the very early stages of the design process. Following that, mathematical models, physical models, block diagrams, specifications, and sketches are used. CAD-tools are used to implement the design.

3.1 Introduction

Many words have been written about the design process ([French, 1985; Horvath, 2000; Pahl and Beitz, 1996] and references contained therein). It is generally accepted that the early part of the design process, the conceptual phase, is relatively inexpensive. The group of people is small, the materials and equipment required are simple and for general purpose. However, the most important decisions are made in that early part, thus defining the cost of the product to a very large extent [Kals *et al.*, 1996, p.17; Lennings *et al.*, 2000; Zuo and Director, 2000]. Therefore it is paramount that the decisions taken in this phase are rigidly founded on firm arguments.

It is expected that models play an important role in the work of conceptual designers. This chapter will look into this by making an inventory of models used, organising them using the model of conceptual design in Figure 2.5 and selecting a set of models that covers the conceptual phase. The results will be used further in this research (see Section 1.2).

As mentioned in Section 2.3, abstraction is widely applied when designing. For abstraction, models are well suited as they provide ways of presenting a relevant selection of the available information. It is therefore expected that they play an important role in the conceptual design phase. A model in this sense, can be defined as follows:

mod·el A schematic description of a system, theory, or phenomenon that accounts for its known or inferred properties and may be used for further study of its characteristics: a model of generative grammar; a model of an atom; an economic model. (From: The American Heritage® Dictionary of the English Language, Fourth Edition Copyright© 2000 by Houghton Mifflin Company. Published by Houghton Mifflin Company.)

This chapter has been published earlier as [Bonnema and van Houten, 2006]. Only minor updates are performed in this thesis.

It should be noted that a model is always a *limited* representation of reality; information that is not relevant to the purpose of the model is left out. Models created for investigating power consumption are therefore not suited for simulating dynamic behaviour.

In the next sections the use of models by conceptual designers is explored. First a list of models from literature on conceptual design projects and from models used by computer support tools for conceptual design is given. Section 3.3 covers the design of a questionnaire to make an inventory of the models used by conceptual designers. This section also contains the results. The results will be further discussed in Section 3.4. Section 3.5 describes the outcome of interviews with conceptual designers on the results. In Section 3.6 the conclusions will be presented.

3.2 Literature on Models in Conceptual Design

3.2.1 Use of models

When we look at different applications of conceptual design in literature [Mekid and Bonis, 1997; Mital *et al.*, 1997; Takasugi *et al.*, 1996; Zurro *et al.*, 1997] we can find many models that are being used:

- Design criteria;
- Graphical model of the environment;
- List of requirements (quantified when possible);
- System budgets (for instance used to divide the total electrical power over the subsystems, or to divide the total allowable positioning error over the subsystems);
- Morphological schemes;
- Kinematic diagrams;
- Sketches of possible solution(s);
- Physics models;
- Scenario for use of the device;
- Dynamic models based on simplified structure;
- Dynamic simulation results;
- Input/output data and data structures;
- Controller layout;
- Hardware/software framework;
- Detail design sketches of critical components;
- Finite Element Models and calculations; and
- Numerical models and simulations.

Though some models seem to be more applicable in detail design, they can be, and have been, used in conceptual design. Finite Element Models, for instance, can be used in the conceptual phase to investigate feasibility without a completely defined geometry.

From the references it cannot be determined exactly what models are used in what part of the conceptual design phase, nor how the models relate to the properties concreteness, complexity and realization (see Figure 2.5), or horizontal, vertical and general abstraction [Kao and Archer, 1997].

3.2.2 Models in conceptual design support tools

As the goal of this research is to create a support tool for conceptual and systems design, it is interesting to see what models are used in existing tools. Four tools are chosen:

- Schemebuilder [Bracewell and Sharpe, 1996];
- 20–Sim [van Amerongen and Breedveld, 2003];
- Dymola [Dynasim, 2005]; and
- CODSAS [Al-Salka *et al.*, 1998]

Schemebuilder — one of the older conceptual design support tools. It was initially directed towards a largely automated design creation system. However, at present it is more aimed at creating a support tool that helps the designer, not replace him. Based on the work of French [1985] it uses the term *scheme* for a concept.

Schemebuilder uses bond graphs as its basis. However, other models that are used in Schemebuilder are (see Figure 1 in [Bracewell and Sharpe, 1996]):

- Yourdon diagrams;
- 3D solid models; and
- function maps.

20–Sim — (pronounce: Twente Sim) was originally developed as a simulation tool. During the years it evolved into a much more versatile program. It provides the designer with a modelling and simulation environment that is also based on bond graphs and the corresponding constitutive equations. The user can see the models in different representations:

- iconic diagrams;
- (electrical) schematics;
- 3D models and animations;
- block diagrams; and
- ideal physical elements.

This approach provides designers of different backgrounds with their own familiar model representations. In [de Vries, 1994] it is discussed how integrity between the different representations can be maintained.

Dymola — is quite comparable to 20–Sim. However, it uses the language Modelica® as its basis. For different applications there are specific libraries with models and elements. The representations of the models to the user can be, just like in 20–Sim, in several forms:

- block diagrams;
- 3D models and animations;
- (electrical) schematics;
- iconic diagrams; and
- ideal physical elements.

CODSAS — (Conceptual Design Support and Analysis System) uses a "new high-level programming language called design procedures programming language (DPPL)" [Al-Salka *et al.*, 1998]. This approach differs greatly from the previous ones. The design process is programmed in a language that provides two data structures (document and integer). The document structure can be in one of six languages (in fact categories of models):

- textual;
- graphical;
- structural;
- hierarchical;
- tabular; and
- morphological.

As the process is programmed, it is indicated which integers and documents (and of what type) are input and output. Not only the models are programmed (in one of the six languages) but also the tasks and activities.

3.3 Models Used by Conceptual Designers

In order to investigate the use of models by conceptual designers further, several conceptual designers have been interviewed using a questionnaire. The questionnaire and the results will be dealt with in this section. The purpose was to make an *inventory* of models and relate their use to the space in Figure 2.5.

3.3.1 Questionnaire design

Based on personal contacts several persons have been identified that can be described as conceptual designers. They typically work in an environment where new designs have to be created (not mainly adaptive design), where advanced technology is applied and where people from several disciplines work together. Important is that they are involved in the early phases of a new design project.

The questionnaire is sent to these experienced conceptual designers by e-mail or given in person. The responses were returned to the author by e-mail, fax or normal mail. To increase the number of responses, the questionnaire is designed to be filled out in approximately 15 minutes. The language is English, which is the second language for a large portion of the target audience. This is considered to be no problem, as the conceptual designers are typically highly educated and expected to be familiar with the English language.

First, a short introduction on the research project is given, including the position of the project in the research program of the Laboratory of Design, Production and Management at the University of Twente. Then practical information and the due date are given. Next, a few questions are asked about the conceptual designer like age and design experience. Then, the working environment and the type of problems the designer is confronted with is asked. The designer is asked to rate the type of problems on a 1 to 5 scale for complexity, concreteness and realization. See the scales shown in Table 3.1.

The larger part of the questionnaire is about the models used. To aid in listing the models, the following categories of models are provided:

1. Textual models, divided into:
   (a) List-like;
   (b) Descriptive; and
   (c) Other textual models.
2. Graphical models, divided into:
   (a) Abstract;
   (b) Concrete; and

Table 3.1: Scales used to rate complexity, concreteness and realization for the problems the interviewees are confronted with.

Complexity:				
Single Device	-	Chain of Devices	-	Structure of Devices
1	2	3	4	5
Concreteness:				
Physical Principle	-	Working Principle	-	Construction element
1	2	3	4	5
Realization:				
Idea	-	Technical Drawings	-	Product
1	2	3	4	5

(c) Other graphical models.

3. Analytical models;
4. Other models.

For each of the models listed, the designer is asked to rate the complexity, concreteness and realization of the corresponding problems using the scales shown in Table 3.1.

Finally, space is reserved for remarks from the interviewee on the questionnaire. Also it is asked whether the interviewee likes to receive more information about the outcome of the questionnaire and/or the research project. The questionnaire is given in Appendix A.

3.3.2 Expected Results

Conceptual designers typically work in the early stage of design, where values on the realization scale are low. On the other hand, conceptual designers should also be involved in the realization phase of a design project to safeguard the chosen concepts (see Chapter 2). It is therefore expected that only a few models for problems with higher realization values will occur.

Also, as conceptual designers are often involved in complex projects, several models for highly complex problems are expected. As mentioned in [Kao and Archer, 1997] abstraction is a very powerful tool for conceptual design. As both the concrete and abstracted problem have to be considered, both high and low values are expected on the concreteness scale.

3.3.3 Results

In Table 3.2 the results of the questionnaire are summarised. The response rate was little over 50%. The designers interviewed are relatively young and have approximately twelve years of design experience, most of which in the field of conceptual design. This may lead to the conclusion that conceptual design is a natural way of thinking for some people who will use it from early in their design career. The fields of industry were: lithography equipment; off-shore and earth moving equipment; industrial design and engineering; and high performance optics. The problems the designers are confronted with are above average in complexity and concreteness, however below average in realization. Several designers mentioned a large range on these three ratings. As can be seen in Table 3.2, a total of 86

models have been identified of which 45 different ones. Where it should be noted that minute differences in the descriptions as given by the interviewees have been removed.

Table 3.2: Summary of results of the questionnaire

Surveys	
Sent	13
Returned	7
Response	54 %
Respondents	
Average Age	37
Average Design Experience	12 years
of which Conceptual Design	10 years
Problems	
Complexity	3.5
Concreteness	3.4
Realization	2.2
Models	
Total number	86
Different	45

One of the respondents gave vague answers to the models he used (only the type or the discipline involved is mentioned). Therefore his answers are left out of the following more detailed analysis. This means that 78 models, of which 43 different ones will be analysed (again with minute differences removed). In Section 3.4 the models mentioned by this respondent will be taken into consideration. The number of respondents (7) is too small to perform statistical analysis. As the purpose was to create an inventory of models, statistics are not needed.

A more detailed analysis of the results is presented in Figure 3.1. Here six plots are shown, each representing one realization value shown above each plot. Within each plot the number of models for each combination of complexity and concreteness is shown. A few models were given no complexity, concreteness and/or realization rating; these are shown with the number zero in Figure 3.1. In addition to that, Table 3.3 shows for each complexity, concreteness and realization value how many models are mentioned in each of the categories. The models mentioned are distributed over all categories.

It is clear that only a few models for problems with Realization ≥ 4 have been identified. This was expected. The models used for problems with a larger realization value were among a few others: Assembly Procedure, Component Testing, Interface Descriptions, and Prototype Testing. These types of models typically occur in the integration phase of a project where the fit between separate modules, both in geometry and functionality has to be checked. The conceptual designer will be involved because of his overview over the entire system (as in Chapter 2). These models have more to do with verification and integration than with design.

For realization = 1, most models identified are relevant for higher complexity and lower

*Table 3.3: Results of the questionnaire showing the number of models in each category as function of complexity, concreteness and realization. The highest number (> 1) in each column is shown in **bold**.*

Number of Models		**Complexity**						**Concreteness**						**Realization**					
	Total	-	1	2	3	4	5	-	1	2	3	4	5	-	1	2	3	4	5
Textual List	12		**2**	2	1	**5**	2	1	**2**	3	2	2	2		3	5	3	1	
Textual Descriptive	14	1	**2**	4	**4**	1	2	1	**2**	**6**	3	2		1		**8**	3	1	1
Textual Other	3			3						2			1			2		1	
Graphical Abstract	14			3	2	2	**7**		**2**	5	**6**	1			**5**	4	4	1	
Graphical Concrete	17		1	**6**	**4**	3	3		**2**	5	3	**5**	2		3	4	**8**	**2**	
Graphical Other	3			2	1					1	1	1				1	1		1
Analytical	8		1	2	**4**		1		1	4	1	2			3	3	2		
Other	7			3	2	1	1			2		1	**4**	1		2	1	**2**	1
Total	**78**	1	6	25	18	12	16	2	9	28	16	14	9	2	14	29	22	8	3

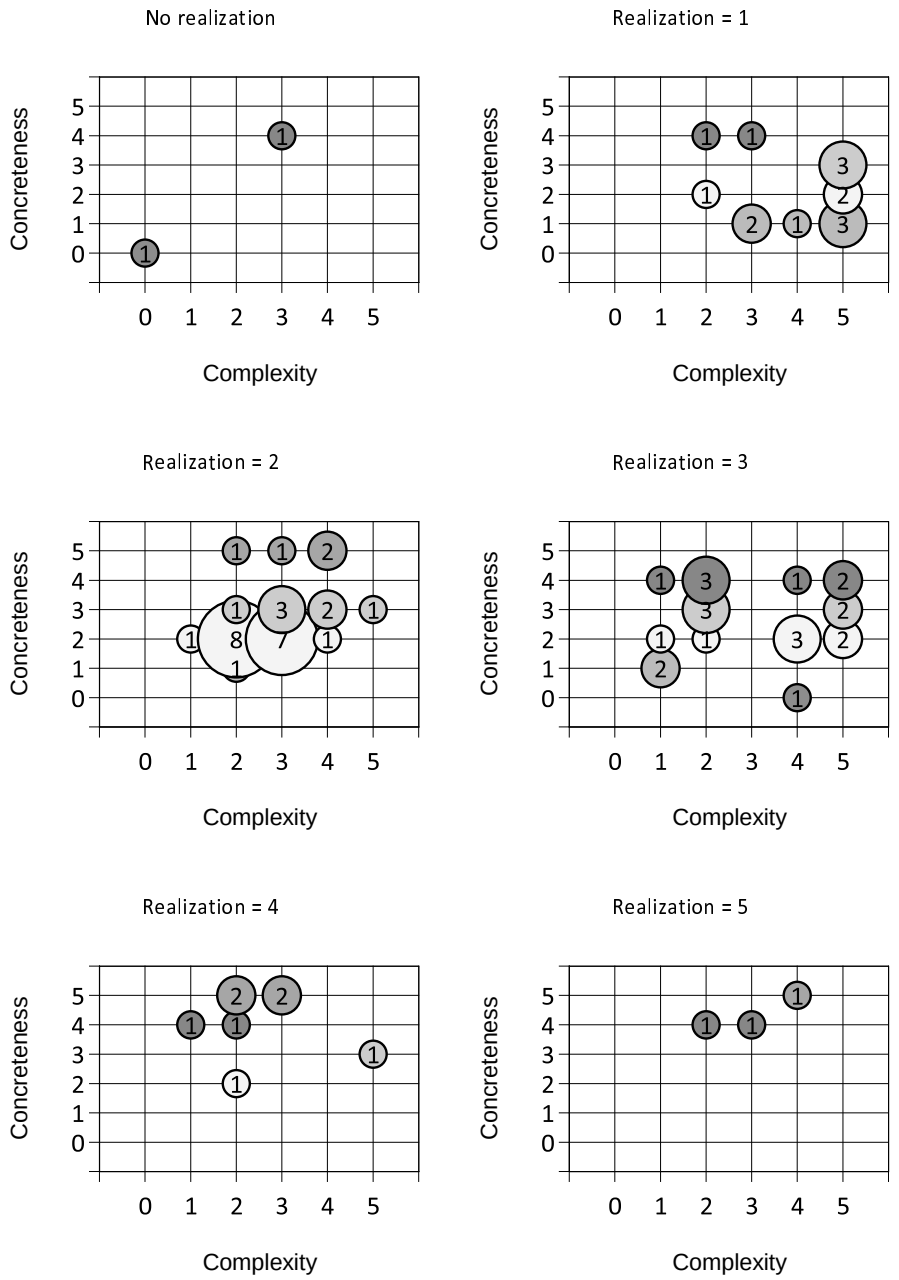


Figure 3.1: Results of the questionnaire. Each plot represents one Realization value (shown above the plot). The horizontal axes represent Complexity and the vertical axes Concreteness. The number of models are indicated by the area of the bubble and the number in the centre of the bubble.

Table 3.4: The models mentioned more than once, together with their frequency (f) and category. The models are shown in order of increasing average realization of the corresponding problems (Real). Also shown are the average values for complexity (Comp) and concreteness (Conc).

			Average		
Model	f	Category	Comp	Conc	Real
Budget	2	Textual List	5.	2.5	1.
Analysis of Physical Behaviour	2	Analytical	3.	1.5	1.5
Mathematical Model	6	Analytical	2.7	2.2	1.7
Block Diagram	3	Graphical Abstract	4.	2.7	1.7
Sketch	6	Graphical Concrete	3.2	2.7	2.
Scenario	2	Textual Descriptive	4.	2.5	2.
Functional Diagram	3	Graphical Abstract	3.3	3.	2.3
Specification	4	Textual List	3.3	3.5	2.5
Description	2	Textual Descriptive	2.5	1.5	2.5
Scheme	3	Graphical Abstract	3.	1.7	2.7
Drawing	2	Graphical Concrete	3.	3.	3.
CAD	7	Graphical Concrete	3.	3.6	3.4

concreteness. The largest number of models correspond to problems with realization = 2. In particular for problems with concreteness = 2 and complexity = 2 and 3, many models have been listed. Looking at Table 3.3, it can be concluded that in particular textual descriptive models are used intensively. Also textual list and graphical concrete and abstract models are used, be it less frequently.

For realization = 3 there is a larger variation for both complexity and concreteness. The total number of models identified for this realization value is a little lower than for realization = 2. By far the largest number of models are in the graphical concrete category. As mentioned before the number of models for problems with large realization values is small.

Table 3.4 shows the most often mentioned models plus their frequency and average complexity, concreteness and realization values of the corresponding problems. (Systematic or Methodical design and Value Engineering have been mentioned several times as well. However, these are design methods, not models. They are therefore left out of Table 3.4.)

The order chosen in Table 3.4 is increasing realization. Referring to Figure 2.5 it can be said that during a design project complexity and concreteness may increase and reduce. Realization, however, only increases (albeit not continuously). Thus ordering the models mentioned by increasing realization values, gives an indication for a logical sequence of models.

The model with the lowest realization value (and thus applicable in the earliest phase of a design process) is a budget. Such a budget is used to divide a system specification over the constituting parts. We will look into this in more detail in Section 5.5. The reported frequency in Table 3.4 is two, but the interviewee whose response has been left out of the analysis, reported intensive use of budgets throughout the design process. Interesting to note is also the high score of complexity for budgets.

Three models with comparable realization values are analysis of physical behaviour, mathematical model and block diagram. Of these, block diagram has the highest complexity value. Analysing physical behaviour is on one hand creating a proper (ideal) physical model

of the situation, on the other hand analysing it with mathematics. These two models are therefore closely related. Note the higher value for concreteness for the mathematical model. To perform mathematics, more concrete knowledge like parameters is needed.

Next on the realization scale are sketch and scenario. Sketch is the first graphical *concrete* model on the realization scale. A scenario describes the use and context of the product to be designed.

The next group of models contains two graphical abstract models (functional diagram and scheme), and two textual models (specification and description). A scheme can be nearly any graphical abstract model, like functional and block diagrams, electrical schematics or timing diagrams (here scheme should not be understood as a synonym of concept, as in [French, 1985]).

Finally two graphical concrete models are mentioned: drawing and CAD. These two are closely related. However, a drawing can be made by hand. It can be a worked out sketch that is used to detail the product using CAD. Its lower realization and concreteness values are therefore logical.

It can be concluded that all models that are mentioned more than once are important for a proper conceptual design process. In the next section the results will be interpreted and discussed further.

3.4 Discussion

This section will discuss the results of the questionnaire into more detail providing the basis for the interviews in the next section and for the conclusions in Section 3.6. Although the response rate is fairly high, the actual number of responses is only seven. Thus, the results have to be considered with caution. In particular it should be noted that statistics are not applicable with this small number of results. On the other hand, as the purpose of this questionnaire was to create an inventory, the number of models mentioned is large enough.

All categories of models are used intensively. Graphical, textual and analytical models are used by conceptual designers throughout the process. For lower complexity problems, both textual and graphical models are used. For high complexity, the graphical abstract and textual list models prevail. Problems with a low concreteness can be handled using models in all categories. As concreteness increases, the graphical models are used more often. For higher realization values, the graphical concrete models play the major role. At lower realization values textual descriptive and graphical abstract are used more often.

As noted earlier, conceptual designers are mostly involved in the early part of the design process, where realization values are low. However, they produce plans for entire solutions, thus both complexity and concreteness will be large at the end of the conceptual phase. Some conceptual designers will be involved in the integration and testing of the final product. They will use models for problems with high realization values.

Let us look at the results of the questionnaire in Tables 3.3 and 3.4 and Figure 3.1 and compare it with the model in Figure 2.5. The path in that model by Krumhauer is to start from high complexity and concreteness, but low realization. Then first reduce concreteness followed by a reduction in complexity. By solving the individual subproblems, the concreteness and realization are increased. Then these individual solutions are combined, thus increasing complexity.

As realization increases throughout the conceptual design process, we can look whether the models mentioned follow the pattern of first reduction in concreteness and complexity, then increase in concreteness and then in complexity.

Figure 3.1 shows that realization = 1 corresponds mainly to models with high complexity and low concreteness. No models are mentioned for both high complexity and high concreteness values. For increasing values of realization, the values for concreteness and complexity vary more. Krumhauer expects for high realization values high complexity and concreteness values. The results of the questionnaire do show models with high concreteness and models with high complexity. However, the number of models for problems with high complexity and high concreteness and high realization is very low.

It appears the model by Krumhauer is not fully supported by the results of the questionnaire. Problems are represented in the early part of the design process by models that are less concrete than Krumhauer suggests. This can partly be caused by the fact that the initial abstraction is performed without models, or that the designers interviewed have a perception of lower concreteness in the early stage of the design. For the higher realization values, where Krumhauer expects high complexity and high concreteness, the results of the questionnaire are neither supportive nor negative.

Looking at [Kao and Archer, 1997], where it is concluded that experts use (vertical) abstraction intensively, we can conclude that indeed the experts interviewed approach a problem at an abstract level. Table 3.3 shows many models for lower concreteness values. Also, the interviewees use horizontal abstraction: the lower complexity values also have many models.

Next let us look at the frequency of models mentioned, and the average complexity, concreteness and realization values (Table 3.4). Again as realization is low in the beginning of the design process and increases during the work of the conceptual designers, a logical sequence of models in the design process is:

1. System budgets for the total view, to handle high complexity and to provide a means for splitting the top level requirements into requirements for the subsystems.
2. Mathematical models for allocating the budgets to (critical) elements. At first rough calculations, moving to more detailed and precise mathematical models. These can also be used for budget verification later on in the design process.
3. Analyses of physical behaviour in combination with mathematical models.
4. Block and functional diagrams (possibly several other schemes) to model the functional dependencies in the design and to maintain overview over the entire system under design.
5. Specifications to record the decisions on the top level design and functional characteristics. Also interface specifications are needed.
6. Sketches to develop possible embodiment details.
7. CAD as a tool to implement and monitor the embodiment and detail design and to create (technical) drawings.

Additionally, sketches will be used to complement and illustrate the other models mentioned above. Also, there should be space to create and maintain descriptive models like scenario's.

The vague answers of one respondent that were left out are not in conflict with this sequence. In fact they support the use of budgets to cope with complexity. Also, the interviews mentioned in Section 3.5 support this list.

Interestingly, the models mentioned above do not correspond to the models used in the conceptual design tools of Section 3.2.2. In particular budgets, specifications, scenarios, and sketches are not available in the four tools. CODSAS may be able to represent budgets as a tabular document. In the description [Al-Salka *et al.*, 1998] they are not mentioned, though. The mathematical models, physical models, block diagrams, and CAD are available in most of the four tools.

The list is not intended as a rigid structure in which the models should be used. Instead, it provides a framework for a support system for conceptual design. Such a system has to incorporate means to handle these models. Equally important is that a system can handle the connections and transformations between these models.

It is worthwhile to mention the research in [de Vries, 1994]. Here three types of models are used in a closely interlinked manner: iconic diagrams, bond graphs and the THESIS modelling language. All three of these models are used for problems low on the realization and concreteness scales. Complexity of the models may be high, but it remains to be seen whether complex bond graphs and THESIS models can be sufficiently easily handled by the designer. Budgets and functional block-diagrams at several levels of complexity and/or concreteness may help in handling complexity. Combined with the principles in [de Vries, 1994] connecting the models mentioned above may be feasible.

3.5 Interviews

The results above have been discussed with three of the conceptual designers in order to verify the results. These designers were from three different fields of industry (lithography equipment; off-shore and earth moving equipment; and industrial design and engineering). In these discussions the kind of design issues the conceptual designers face have been discussed, together with their approach. Next, the model by Krumhauer (Figure 2.5) was treated, including the models they use in the different steps (Abstract, Divide into Subproblems, Search for Solutions and Combine and Select). Finally, the list of models for the conceptual phase (Section 3.4) was discussed.

The design problems described varied considerably, but in all cases a complex context and a complex, multi-disciplinary solution were mentioned. The approach consisted in all cases of a structured method, based on, or equal to systematic design [Pahl and Beitz, 1996]. One interviewee also mentioned the use of TRIZ [Altshuller, 1997, 1999] as an approach to the problems he faced.

There were several interesting remarks on the model by Krumhauer. In general the route though the design space was recognised. However, iteration is not shown in the model, but is present in reality according to the interviewees. Also, in case of extreme specifications, it is important to investigate the feasibility in an early stage of the design process. This cannot be derived from the model by Krumhauer. Also, as mentioned in Section 3.4, the first abstraction step in the model was less recognised. One interviewee mentioned that in his opinion the goal is often present *before* the requirements are defined. The models used in the four steps of the model by Krumhauer were a subset of the ones mentioned in the responses to the questionnaire.

The main result of this chapter is the list of models presented above. As said, this list has been discussed in the interviews. The question was whether the models were recognised in approximately this sequence. One of the interviewees did not recognise the use of system

budgets for errors, performance, power use etc. However, the use of a space budget in a product with a strict size constraint was used. Another interviewee mentioned the fact that a budget can only be made when a functional model is already created. This asks for an integrated approach to these two models. Two of the interviewees did not need CAD in the conceptual phase. It was considered as a tool for detailing the product. One interviewee even considered the use of CAD highly overrated.

All the models in the list in Section 3.4 were considered important. Additional models that may be required were patents, comparisons to other products, and an integrated system for combining mechanics and electronics.

3.6 Conclusions

As conceptual design is the phase in which different fields of science and practical knowledge have to be handled in order to define the basic operation and structure of a new product, a large amount of information has to be handled. To accomplish this, models are used by the designers. Each model is able to represent and manipulate a specific combination of information. To investigate the use of models by conceptual designers a small-scale literature study is made. Different models are listed, without the possibility to link them to the model presented in [Krumhauer, 1974].

A questionnaire has been designed that is used to make an inventory of the models used by conceptual designers. Also the complexity, concreteness and realization of the problems for which the models are used are asked. This enables linking the models to the model by Krumhauer. The questionnaire was sent to experienced conceptual designers; the response was 54%.

The results of the questionnaire show that the designers interviewed have started doing conceptual design shortly after having started designing. They also show that conceptual designers use many different models throughout the design process. Each combination of complexity, concreteness and realization asks for a specific type of model to obtain and maintain insight and overview. A possible sequence of models has been identified that can be used in a support system for conceptual design. However, there is no information yet on the connections between the models in this sequence. It is worthwhile to investigate that further, as a system that uses these connections can help in investigating the consequences of design decisions early in the process. Also, such a system may help in guiding the design process.

The models provide a means for handling the information at different stages of the conceptual phase. They can be used for communicating and analysing problems. For solving the problems, other techniques and methods are required. One can think of traditional methods like brainstorming/brainwriting and morphological schemes [Pahl and Beitz, 1996]. A very powerful method is TRIZ. TRIZ helps to formulate the problem, but also provides tools (40 principles, contradiction matrix, evolution of technical systems, substance-field analysis) that guide the designer to several solutions.

The model by Krumhauer is not supported by the results of the questionnaire but in the interviews the conceptual designers recognised the route through the design space. The reduction in concreteness at the beginning of the conceptual phase is not recognised in the models listed by the designers. In the interviews this step was the least recognised. The increase in complexity at the end of the conceptual phase cannot be found in the results

either. However, this does not mean the model by Krumhauer is incorrect. Possibly the reduction in concreteness is performed by the designers without the use of models. The same may hold for the increase in complexity.

As complexity is a characteristic of the problems conceptual designers are confronted with, it is paramount that models are used that can handle these highly complex issues in the earliest stages of the conceptual phase. The models for these issues that are listed in the questionnaire are budgets, block and functional diagrams, and specifications. These should therefore be included in our system design support system. As a backbone, a mathematical approach should be considered. Representation of the systems using iconic diagrams or ideal physical elements would be convenient as well.

This chapter has only looked at the most frequently mentioned models. However, it is expected that by using the models in Section 3.4 most critical issues in conceptual design can be handled. In the next chapter the models found in this chapter will be used to derive a reference model for concept and system design.

Chapter IV

Conceptual and System Design Reference Model

From the study of the design process, and particularly the conceptual phase, in Chapter 2 and the results of the investigation into the use of models in Chapter 3, we will derive a reference model for the conceptual and system design phase. For this we will investigate complexity, concreteness and realization further. The reference model can be used to find similarities and dissimilarities between the different design processes. We will map several of the design process models in Chapter 2 in the reference model. This shows how the different process descriptions can be connected. Also, the requirements for a system design support system will be derived.

4.1 Introduction

In Chapter 2 several models of the design process and in particular the conceptual design phase have been shown. Some describe the process in steps and actions to be performed. Others used the data involved to describe the process. It is hard to compare these different descriptions. The present research does not only incorporate the conceptual design phase, but also aims at supporting system design and system architecting. Therefore the design process models taken from literature and presented in Chapter 2 require combination and extension to incorporate these aspects.

In this chapter we will derive a reference model for conceptual and system design that can be used to map process driven models and data driven models. The (product) models used by designers, reported in Chapter 3, provide information on what aspects need to be covered by such a conceptual and system design reference model.

4.2 Krumhauer Revisited

Up to now, the model in [Krumhauer, 1974] (see Figure 2.5 on page 17) has been used to describe the conceptual design phase. This model, developed before 1974, describes the conceptual phase in a space defined by three axes: complexity, concreteness and realization. In addition to this space, the route through this space during the conceptual design phase is given. Krumhauer has a prescriptive approach to conceptual design; the steps to be performed in the conceptual phase are clearly given. It is therefore closely related to the approach presented in [Pahl and Beitz, 1996] (see Figure 2.7 on page 21). In Chapter 3 it was concluded that the interviewed conceptual designers did recognise the space and the general route as prescribed by Krumhauer.

Krumhauer prescribes a start from a list of requirements, then to abstract to come to the goal of the project. The designers interviewed expressed their view that it is often the other way around: the list of requirements is derived from the goal. In general, an interaction between goal development and requirements development will occur.

Krumhauer shows a one-way route through the space. Although that is the most desirable route, in practice iteration will occur. Thus, a new reference model must allow for iteration. It should be noted that iteration is repeating steps with improved starting conditions in order to find alternative or better results. Related, but not equal, to iteration is recurrence where the same *type* of steps are performed, but at an increased level of detail. Iteration thus is repetition at the same level of the hierarchy, while recurrence is repetition at another lower level.

The basis formed by the space can be reused for the new model, although the description of the axes is not immediately clear to everyone. Finally, the fact that system design is a highly hierarchical process is not accounted for in the Krumhauer model.

4.3 Hitchins, Alexander, French, and CAFCR Revisited

The models by Hitchins [1992], Alexander [1966] and French [1985] and the CAFCR model [Muller, 2004b] do not focus on the conceptual design phase, but look at the design of systems. In particular the model by Alexander [1966] shows distinct steps of mental reasoning and formalisation. This conforms to the observations in [Kao and Archer, 1997] where wide application of abstraction is said to be used by expert designers. Thus in each step of the process, abstraction should be supported in the reference model.

Sometimes a formal approach is needed to perform abstraction. In particular when treating physics, the formal mathematical approach is required to analyse the phenomenon at hand. In [Muller, 2004b] the informal side of designing systems is treated. The informal side is necessary because the problem and solution spaces are too large to be be sampled completely. It is required to sample only the critical subspaces. [Muller, 2004b] proposes to ask many WWHWWW questions (why, what, how, who, when, where) and to sample the space using this simple rule:

> How about the `<characteristic>`
> of the `<component>`
> when performing `<function>`?

As a balance, the results of such informal sampling should be (somewhat) formalised for the design record and to be used for further design decisions.

[Hitchins, 1992] shows design as a closed loop. Every time a resolution to issues is created, new issues will emerge. Design is about addressing and resolving such issues. Related to iteration, it is required that the model supports these kind of loops.

4.4 Requirements on the Reference Model

From the above, the following requirements on the conceptual and system design reference model can be derived:

- Use a space like the model by Krumhauer;
- Use clearer descriptions for the axes;
- Allow for iteration;
- Allow for abstraction;
- Allow for formal and informal approaches;
- Allow for different routes through the design space;
- Allow for hierarchical designs (and thus allow for recurrence).

4.5 The Reference Model Space

The space introduced by Krumhauer will be reused as it was recognised by the designers interviewed. A problem is posed by the three axes defining the space. Krumhauer only uses three terms: complexity, concreteness and realization (see Section 2.3). The designers indicated a clearer marking of the axes is necessary. Let us therefore investigate the three scales further.

Please note that the space is not used to classify a design problem, but to visualise the route taken during the design process.

4.5.1 Complexity

As the reference model has to be applied on designing complex systems, a further elaboration on *complexity* is appropriate. Complexity is widely studied (see for instance [Axelsson, 2002; Casti, 1979; Gell-Mann, 1995; Manson, 2001; Martensson, 1999; McDermid, 2000; Nakagawa and Yasui, 2003; Ottino, 2003; Schön and Bennet, 1996] and also see Section 2.2.2). It is not our object to repeat these.

In normal conversation the adjective "complex" is used for two phenomena:

- A *difficult* problem;
- A problem or system that has properties and behaviour one cannot directly oversee.

From literature we can find more information on this. Krumhauer only related complexity to the number of objects. Referring to [Manson, 2001] three types of complexity can be distinguished:

- *Algorithmic complexity*, in the form of mathematical complexity theory and information theory, contends that the complexity of a system lies in the difficulty faced in describing system characteristics.
- *Deterministic complexity* deals with chaos theory and catastrophe theory, which posit that the interaction of two or three key variables can create largely stable systems prone to sudden discontinuities.
- *Aggregate complexity* concerns how individual elements work in concert to create systems with complex behaviour.

[Pugh, 1991] deals with complexity explicitly. It is defined as a number:

$$C = \frac{K}{f} \times \sqrt[3]{N_p N_t N_i} \tag{4-1}$$

Where:
- K: scaling factor;
- f: number of functions to be performed;
- N_p: number of parts;
- N_t: number of different types of parts;
- N_i: number of connections and interfaces.

This clearly shows that complexity is related to the number of parts, and their different types, and the connections and interfaces, as also reported in [Deshmukh *et al.*, 1998] for manufacturing systems. However, these parameters are not independent: when the number of parts increases, the potential number of connections increases progressively [Hitchins, 2000].

Table 4.1: Detail and Dynamic complexity according to [Calvano and John, 2004]

Detail complexity	**Dynamic complexity**
Weakly-integrated systems	Highly-integrated systems
• Hierarchical relationships dominate lateral influences. • Cause and effect are relatively obvious and direct. • The implications of design decisions are relatively predictable. • Risks are dominated by the local risks and achieving the contributing parts. • Influences on, and implications of, decisions tend to follow the local partitioning of the solution elements.	• Lateral influences dominate hierarchical relationships. • Cause and effect are not obvious and direct. • The implications of design decisions are much less predictable. • Risks are dominated by system risks, with unforeseen emergent properties. • Influences on, and implications of, decisions are much more difficult to bound and to establish.

In Figure 1.1 on page 4, a pyramid is shown that relates the number of items to different aggregation levels. At the bottom there is an enormous number of things to consider. Once the items have been divided in relatively independent chunks, finding solutions is relatively straightforward. It does require a significant amount of work, though. Looking at the top of the pyramid, we see a limited number of issues. Nevertheless they are entangled and closely coupled. Only by analysis, the exact relations can be found, and subsequently a decoupling can be created. The pyramid can be used for all three forms of complexity defined above. The types of complexity show up in the intermediate level of the pyramid.

Another view on complexity is given in [Calvano and John, 2004] where two types of complexity are identified, also see Table 4.1:

- Detail Complexity: complexity that occurs in weakly-integrated systems.
- Dynamic Complexity: complexity that occurs in highly-integrated systems.

Calvano and John contend that dynamic complexity is much harder to deal with than detail complexity. In particular the identification and mitigation of risks is much more difficult. Only an improved understanding of the nature of integrated systems can reduce the risk level.

[Axelsson, 2002], basing on [Gell-Mann, 1995] and [McDermid, 2000], gives the following aspects of complexity:

- Scale: the number of elements in the system;
- Diversity: the extent to which the system is made up of different elements;
- Connectivity: the inter-relationships between the components;
- Optimisation: the amount of fine-tuning involved in selecting the element's parameters.

The first three items have been described above using different wording. The last item, optimisation, is a new addition. It involves on the one hand the effort required to find a working solution when the problem at hand is so difficult that any solution meeting the requirements is satisfactory. This, for instance occurs when designing a wafer stepper with sufficient overlay performance. On the other hand, it may deal with the effort of finding an optimized solution that maximizes, for instance, revenues for the user of the system. Like a truck that is more economical than another one.

Encapsulation is a term from software engineering, in particular from the object oriented software engineering method [Mattos *et al.*, 1993]. Here, encapsulation is described as when access to attributes of an object is possible only via methods that apply to the object. In more general terms, we can describe encapsulation as hiding details of an object, but exposing the necessary behaviour.

This can also be observed in electronic engineering. Symbols are used that describe a conglomerate of components that perform a certain function (for instance a lying triangle for an operational amplifier (OpAmp), see Figure 4.1). There is a drawback to this approach: if a parasitic effect occurs in the encapsulation, it will not be noticed outside the encapsulation. This is the reason electronic engineers have to understand how an OpAmp works, before they can safely use that component in a real-world design. For the same reason it is considered bad practice in software to modify global variables from within a procedure, as this may result in unforeseen effects.

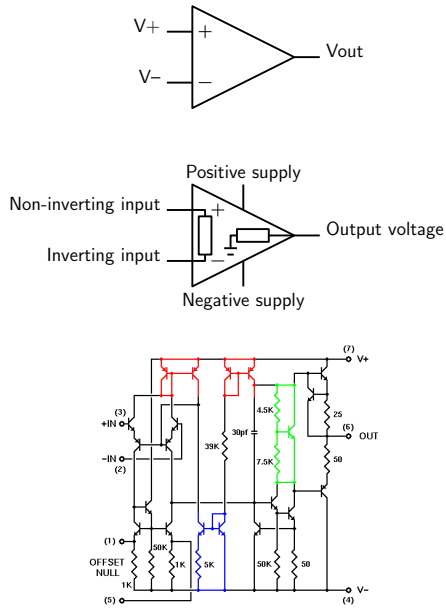


Figure 4.1: Encapsulation of an OpAmp. The top image shows a fully encapsulated scheme ("black box"), the middle image exposes some of the non-ideal behaviour ("grey box"), while the bottom image shows the internals of an OpAmp.

In systems design, encapsulation may be applied in the way that certain pieces of functionality are grouped and treated as a "black box". Often such a black box carries the name of its main function ("power supply", "position sensor", etc.). This reduces the number of individual components to take into account and therefore helps to keep the big picture. This approach is useful as long as the limitations, in particular side-effects, are known. While the design project moves forward, each black box will be filled in with new, smaller, black boxes or real-life components. This inevitably produces side-effects and parasitic effects, and modified demands on the infrastructure. The problems that are associated with these effects are in general not communicated to the next higher hierarchical level. Using a "grey box" that does expose some of its internals would be a better approach.

It is therefore worthwhile to have the possibility to show partitioning of the system into black/grey boxes (the system architecture, see Chapter 1), while on the other hand channelling information from the lower hierarchical levels (the bottom of the pyramid in Figure 1.1 on page 4) to the system level (the top of the pyramid), see Figure 4.2.

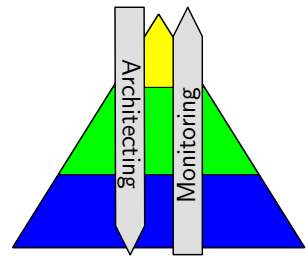


Figure 4.2: The creation of architectures and monitoring of the subsystem designs. Also see Figure 1.1.

4.5.2 Concreteness

According to [Krumhauer, 1974] and the way we have used the concreteness scale in the questionnaire (Chapter 3, Appendix A), the scale covers the range from physical principle, which is an abstract entity, to construction element, which is a concrete entity.

Note: the process from system design to the actual system that is deployed, relates to the realization scale treated in Section 4.5.3, below.

The Krumhauer model starts from a concrete problem. That is not applicable in designing complex systems, as already mentioned by the designers interviewed in Chapter 3. The designers mentioned that the early part of the design process starts from a goal and specifications are derived from that, whereas Krumhauer prescribes the opposite route. Finding the goal is sometimes part of the design process, in particular in the area of industrial design (engineering). There a request for a new product is made and the first part of the project covers this goal-finding [Eekels and Poelman, 1995; Eger *et al.*, 2004a]. We will not look further into that. However, insight in the stakeholders is required to define the correct product, in addition to correctly defining the product. Later the key drivers will be introduced to express the stakeholder's interests (see Chapter 5).

[Kao and Archer, 1997] contend that abstraction is an important way of approaching a problem. Experts solved the problems faster by abstracting as much as possible. [Kroonenberg and Siers, 1992; Pahl and Beitz, 1996] advocate the use of morphological schemes in finding solutions for the individual functions. The best way to use such a scheme is by making sketches of the solutions; not only describing the solution in words. This implies some concretization of the idea. Combining these two, it can be said that frequent abstraction and concretization will provide for increased insight. A concrete solution can be used as seed for finding related solutions. These, in turn, can be abstracted to find for instance the physics that enable them. An abstract solution has to be made concrete to investigate its feasibility. Thus concretization and abstraction will occur often in the design process, just like creating architectures and monitoring progress in Figure 4.2. This has to be supported by the reference model.

4.5.3 Realization

In Chapter 3 we used the realization scale (from idea to product) to track the progress in a design project. The models listed by the designers have been lined up corresponding to their realization score. The realization scale could therefore be seen as the time line of the design project. There is more to it. The realization scale can also be used to regard the life cycle of the system, including its creation, production, usage and discarding.

The well-known "Vee"-process model [Blanchard and Fabrycky, 1998] shows how realization comes into system design, Figure 4.3. On the system level, the system-level requirements are defined, together with a distribution into subsystems. Directly linked to these requirements, test specifications are compiled that will be used for qualifying the final system. On each level in the design process, going downward in the Vee, the coupling between creation and qualification is made by compiling the test specifications.

In system and conceptual design the interaction between the system design and its production are relevant: the system designer has to consider all phases of the product's life cycle in every decision as shown in for instance the Vee-model. In the reference model, we

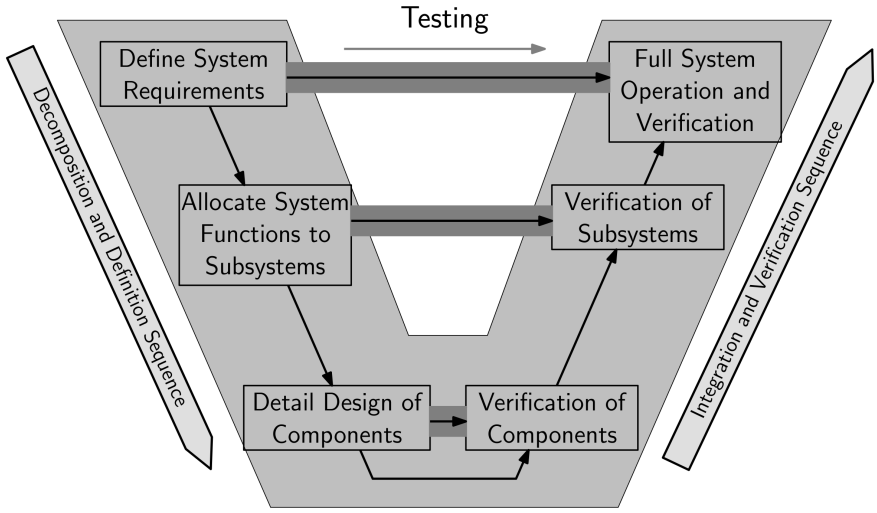


Figure 4.3: The "Vee"-process model for systems engineering [Blanchard and Fabrycky, 1998].

will not use a realization scale, but a scale that shows the progress from vague idea to actual product.

4.6 The Reference Model

Based on the above, we will modify the three scales into bipolar scales:

- Simplex ↔ Composite;
- Abstract ↔ Concrete;
- Mental ↔ Actual.

In addition to these scales based on Krumhauer, another aspect is relevant: Formal ↔ Informal. This poses a problem as four dimensions is beyond the normal human mind. On the other hand, this "scale" relates to the methods used to move within the conceptual and system design space. We therefore do not show a fourth axis, but add that all routes can be traversed using formal and informal methods. Often a designer will alternate between formal and informal approaches to achieve his goal.

Figure 4.4 shows a visualisation of the reference model, where the space is visualised as a cube. A sphere is an alternative, but is not chosen for correspondence with the Krumhauer model. The three axes are perpendicular, representing independent scales.

Please note that the model provides for iteration and design loops. Yet it does not prescribe a certain route through the design space. It allows for description of specific parts or phenomena of designing by regarding only one of the faces of the model. This means detail research on certain parts of the design process can use this model as well.

Also note that both data-based models, like the CAFCR model (Figure 2.4) and the six-box TRIZ model [Nakagawa, 2006], and action-based models, like the models by Pahl & Beitz (Figure 2.7), can be visualised.

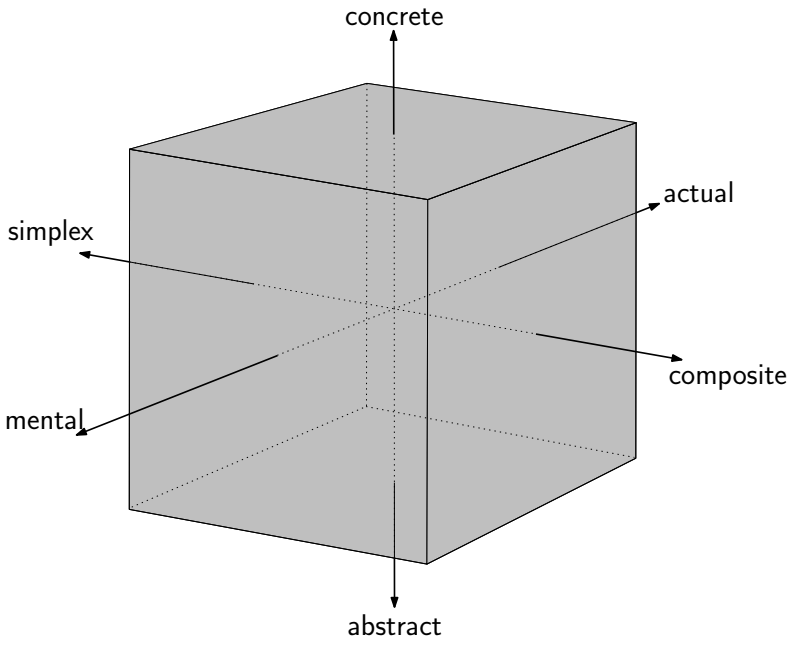


Figure 4.4: The conceptual and system design reference model.

4.7 Design Process Models and the Reference Model

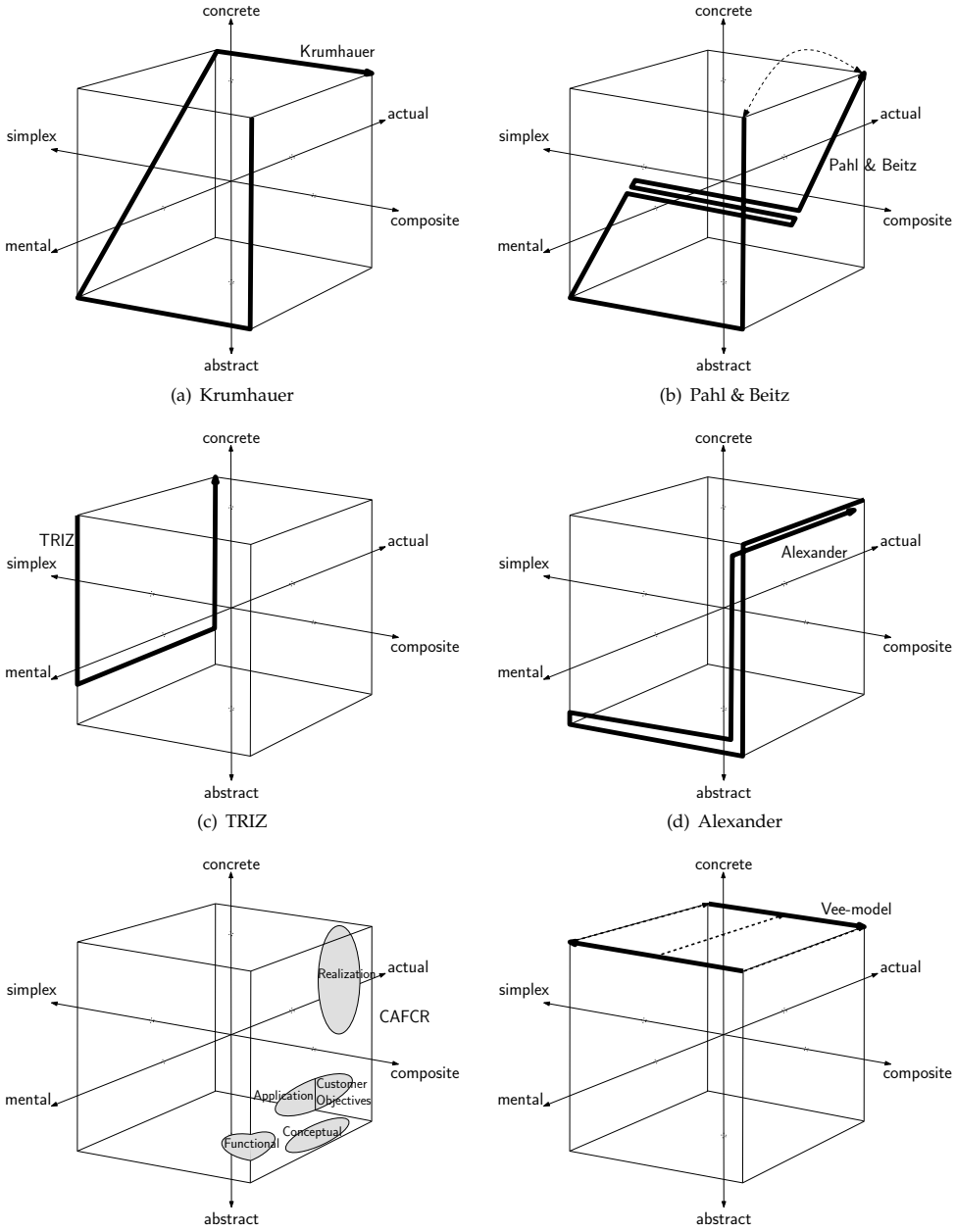
Now that we have a reference model, it is interesting to map different conceptual design process models on the reference model. We will look at the following models:

1. Krumhauer, Figure 4.5(a);
2. Pahl & Beitz, Figure 4.5(b);
3. TRIZ, Figure 4.5(c);
4. Christopher Alexander, Figure 4.5(d);
5. CAFCR, Figure 4.5(e);
6. Vee-model, Figure 4.5(f).

Process models 1 to 5 have been treated in Chapter 2. Other process models treated there do not have conceptual or systems design as subject. Thus they will not be mapped on the reference model. Process model 6 is treated above. In the following we will only visualise the routes. The distinct steps and intermediate results are not shown.

4.7.1 KRUMHAUER

The route by Krumhauer (Figure 2.5, page 17) through the design space can be directly copied. It is striking that concretization and actualisation occur simultaneously.



(a) Krumhauer (b) Pahl & Beitz

(c) TRIZ (d) Alexander

(e) CAFCR (f) Vee-model

Figure 4.5: Different design process models in the conceptual and system design reference model.

4.7.2 Pahl & Beitz

The model by Pahl & Beitz (Figure 2.7, page 21) has to be analysed in order to connect it to our reference model. Fortunately some of the terms in the steps of the model directly correspond to one of our axes.

The starting point of this model (specification) is, like the previous one, concrete and composite, but not yet actual. A specification is only mental, not actual. Then abstraction is proposed, so we move downward in our reference model. Then function structures have to be defined that relate the overall function(s) to subfunctions. This equals simplifying the problem by reducing complexity. The composite function blocks are reduced to less composite, ideally simplex, blocks. Then suitable solutions to the subproblems are sought. This means moving in the direction of concrete, and actual. Up to here, the route equals the route by Krumhauer.

In the next step Pahl & Beitz will deviate from Krumhauer. The former namely prescribes the combination and selection of suitable combinations of working principles. This results in a zig-zag like movement along the simplex–composite axis. Figure 4.5(b) shows three zig-zags, but any number can occur; also along other points of the route through the simplex plane. Eventually, a suitable combination will be selected and worked out. The dashed double-headed arrow in Figure 4.5(b) represents the evaluation.

4.7.3 TRIZ

TRIZ is seen as a conceptual design method. TRIZ by itself, though, cannot deal with composite problems (this relates to complex in the meaning of deterministic and aggregate complexity; TRIZ is *very* suitable for algorithmic complex (or: difficult) problems). We nevertheless show the TRIZ route in the design space, as TRIZ will be incorporated in our system design method presented in the next chapters.

Characteristic for TRIZ is a detour, see Figure 2.8 (page 22). Instead of trying to solve the problem at hand, it is first translated into an abstracted, generalised formulation. TRIZ preferably uses a problem formulated as a contradiction. Then TRIZ provides several tools to solve the generalised problem with a generalised solution. The final step is to transform the generalised solution into a specific solution.

The consequence of the fact that TRIZ does not apply to composite problems, is that the TRIZ route only uses the "simplex" face in the reference model. The detour described above is clearly visible in Figure 4.5(c). This detour is present in both the four-box (Figure 2.8) and the six-box representation [Nakagawa, 2006]. Both are visualised in the reference model by the same route in Figure 4.5(c). This due to the fact that the four-box scheme shows the processing steps, whereas the six-box scheme shows the data involved, that result from the processing steps.

4.7.4 Christopher Alexander

Although the model by Alexander [1966] is not a real design process model, the issues that it illustrates are important to consider (see Figure 2.2, page 13). We can model the situation in the complex, self-conscious situation (Figure 2.2(c)) by the scheme in Figure 4.5(d). The move along the bottom left edge and back is not taken from the model in Figure 2.2, but from the text in [Alexander, 1966]: The solution for the described problem is to split the requirements

into sub-requirements in a tree. This means a decomposition in more simplex issues. After having found solutions for the subproblems, the design process runs in opposite direction along the same route.

4.7.5 CAFCR

The Customer Objectives, Application, Functional, Conceptual and Realization (CAFCR) model presented in Figure 2.4 (page 16) does not describe a process and transformations. It only models the type of information that has to be processed. The CAFCR model regards mostly abstract information, and does not aim to define simplex problems. It provides insight by helping the designer to concentrate on the proper type of information.

The boxes of the CAFCR model can be positioned in the reference model as shown in Figure 4.5(e). Note that they are all placed in or near the composite face of the model.

4.7.6 VEE-MODEL

The Vee-model shown in Figure 4.3 can also be mapped in the reference model. As the left branch of the Vee-model shows how to move from system via subsystems to components, the route in the reference model space goes from the composite to the simplex face. The right branch of the Vee-model represents the integration from components to subsystems to systems. The test criteria directly follow from the corresponding module specifications. In the reference model we therefore see two arrows in the top face, with dashed arrows representing the derivation of test criteria, connecting the two arrows.

4.8 Conclusions

We can conclude that the reference model provides a tool for visualising different approaches to the system and/or conceptual design process. It should be noted that any move in the space defined by the model can be made using very different tools, or according to various methods. These can be either formal or informal. Returning to our goal (Figure 1.2, page 8), we can see that in order to maintain overview, we have to move in parallel to the simplex $\leftrightarrow$ composite axis. To create insight, as suggested by [Kao and Archer, 1997; Pahl and Beitz, 1996], abstraction and subsequent concretization is required. This corresponds to moving in parallel to the abstract $\leftrightarrow$ concrete axis. Innovation is not visible in the reference model. We will see in Chapter 6 that TRIZ is a valuable addition to achieve innovation.

We have shown the routes and data described by different design process models. Later we will see the route we propose. It is interesting to see that the models do not agree on the starting position of the conceptual design phase. Krumhauer, Pahl & Beitz and the Vee-model agree that the starting position is composite, concrete and mental. Alexander has an actual, composite and concrete starting position because he starts with analysing the context of the design problem.

TRIZ differs because it is strong in solving difficult problems, not composite problems. CAFCR has no clear start and end. It merely models the data involved in system design. The Vee-model does not provide answers on *how* to create the system.

If a new tool is designed that supports the concept and system design phase, this tool must provide for abstraction and concretization. Not only as two distinct steps in the

process, but also as a way of approaching the problem at every hierarchical level. The entire design process should be incorporated, because the (system) designer will use his model of the product also during the integration and test phases, as seen in the Vee-model. Most importantly, it should help the system designer(s) in disentangling the complicated design problem so that it can be solved by loosely coupled design teams that are sufficiently informed of the context of their problem. Simultaneously, it should inform the system designer(s) of the progress and consequences of solutions found by these design teams. All this has to be done without trying to reduce the design problem to a basic exercise. Design of complex systems will always be difficult, and will require human intelligence and reasoning capabilities.

The next chapters will be used to create and evaluate such a system design support tool.

II Supporting Design of Complex Systems

Chapter V

Function and Budget Based System Architecting

The chapter presents the principles of a method that integrates functional modelling and requirements modelling to support the system architect in creating and comparing system architectures. Integration is achieved by a coupling matrix that connects functions to key drivers or requirements. Using the matrix, the functions found are allocated to subsystems to create architectures. The coupling matrix is subsequently used to generate budgets for distributing requirements over the subsystems. Architectures can be created easily. Therefore comparison is facilitated. Axiomatic design may be used to create metrics for this comparison.
The elements of the coupling matrix can consist of crosses or ones early in the process to indicate what functions contribute to what requirements. By interviewing experts, the contributions of functions can be estimated more precisely, either by using values, or symmetrical triangular fuzzy numbers (STFNs). When the budgets are created and detail design starts, the specialist designers can compare their achievements with the budget. The system designer(s) can track progress on the system level.
The method provides possibilities of incorporating TRIZ using the 39 parameters in combination with the contradiction matrix. This way, the method is not just an analysis tool, but a design tool: it will actually give clues to how the system to be designed can be improved.
In addition to the principle of the method a possible implementation is presented together with two examples.

5.1 Introduction

This chapter will first look at related methods in Section 5.2. Then the functional view on a system is treated in Section 5.3. Here, several models are presented, based on examples from literature and on the results of the interviews in chapter 3. As we have seen (Section 1.3), a system architecture also distributes the performance over the constituting parts (see page 5), next to the functions, we will look at the performance view on the system in Section 5.4.

The two views are integrated in Section 5.5 where the method is proposed and illustrated with two examples (Subsection 5.5.4). Conclusions in Section 5.6 end this chapter.

5.2 Related Methods

[Bustnay and Ben-Asher, 2005] propose a method of partitioning a system, based on the well-known N^2 method [INCOSE SEH Working Group, 2000]. Their starting point is when the interfaces have been clearly defined. We like to propose a method that can be used *before* the interfaces are identified. In fact, it will help determining them.

Axiomatic design bases on two axioms [Suh, 1990, p.9]:

1. *The Independence Axiom*: Maintain the independence of functional requirements.
2. *The Information Axiom*: Minimize the information content.

This chapter is a reworked version of [Bonnema, 2006a].

These two axioms provide a powerful tool for evaluating designs and design decisions. Though there are similarities to the method presented in this chapter, the most distinctive difference is that axiomatic design focuses on functional requirements and design parameters; these design parameters being parameters describing the solution. The method presented here focuses on (functional) requirements, functional relations and system partitioning. Also, the present method aids the designer and points to possible solutions and architectures. This means the design parameters need not be known for this method to be applicable. Axiomatic design is an evaluation tool to rate alternative designs.

Quality Function Deployment (QFD) is a technique to incorporate quality into products from the outset of the design project. It originates from Japan in the late 1960s and was developed by professors Shigeru Mizuno and Yoji Akao [QFD Institute, 2005]. [Govers, 1996, and references contained therein] describes the fundamentals and application of QFD. Although the technique is very comprehensive, the distribution of the functions over the system's parts is not treated. The present method may thus complement QFD.

Another attempt has been made in [Alexander, 1966] (also see Section 2.2.2). There the *requirements* are arranged in a tree structure. Possible connections between requirements are expected to lead to a solution. As mentioned in Section 2.2.2, the requirement space is so big that it cannot be comprehended.

The software architects at FEI (`www.fei.com`) have created a description of their software architecture in [Driessen *et al.*, 2006]. In this description, a scheme is shown (Table 1 in the reference) that closely resembles the present method. However, instead of coupling functions to key drivers, as we do, the way the key drivers (in the columns) develop into functional requirements (in the rows) is illustrated. In fact the requirements definition process can be visualised and organised in this way; thus the present method and the one in [Driessen *et al.*, 2006] complement each other.

Finally the closely related budget-based design method presented in [Frericks *et al.*, 2006] has to be mentioned. Budgets are here the main model in the system design process. The decomposition in the budget is based on the decomposition of the system under design. The reference (p.64) poses that for the budget the system decomposition "that most closely corresponds to the budgeted resource" has to be chosen for the budget, not the "most obvious functional blocks". The method proposed in this thesis, that will be elaborated in this chapter, follows the budget-based design method in [Frericks *et al.*, 2006], and integrates it with the functional view on the system. In doing so we will align the decomposition based on the budget and the decomposition based on functions.

This chapter will explore the functions of a system as the leading item in creating a system architecture. The functions will be used to define and reuse system budgets and to create a system architecture.

5.3 Functional View on a System

As can be seen from the interview results in Chapter 3, the functions to be performed play a central role in the early phase of design. The form of the product is less important; also see [Eekels and Poelman, 1995; Eger *et al.*, 2004a; Malmqvist, 1994].

In software engineering, the functional description is used for this: what does the computer program have to do? These techniques can also be used for design in other fields of engineering, and when a combination of these fields is required. The descriptions can take

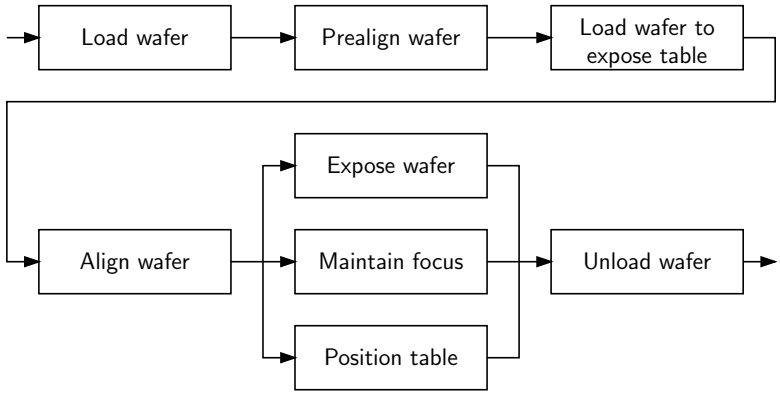


Figure 5.1: Functional block diagram of a wafer stepper/scanner.

several forms. In this section several ways of analysing and representing these functions will be treated that are partly based on the findings from Chapter 3:

- functional (block) diagrams;
- story telling;
- state transition diagrams;
- influence diagrams; and
- mindmaps as function trees.

When developing complex systems, there is not one type of functional model that suffices. While developing one diagram, another is improved, and the need for a third type of model arises. The models treated next should be regarded as the contents of a tool box, where one can select the most appropriate model, or combination of models, for the task at hand.

5.3.1 Functional block diagram

A functional block diagram (FBD) not only lists the functions, it also provides the time-sequence. In addition to the functions, details on the interfaces between adjacent functions can be given. This should not be done too early, as the proper choice of groups forming sub-systems may alter the required interfaces significantly. As an example, look at the functional block diagram in Figure 5.1. It shows the top level functions of a wafer stepper/scanner (also see [Muller, 2004c]). The following functions are derived from Figure 5.1, but can easily be derived from other models as well:

- Load wafer;
- Pre-align wafer;
- Load wafer to expose table;
- Align wafer;
- Expose wafer;
- Maintain focus;
- Position table;
- Unload wafer.

Functions can be aggregations of more detailed functions, e.g. pre-align wafer consists of rotate wafer and find notch. As systems design is often performed on different hierarchical levels, the analysis of functions has to be carried out several times, lower in the hierarchy and with more detail. The results of these lower level analyses and design decisions have to be communicated to higher levels. This can be achieved by detailing each of the blocks in

the functional block diagram (Figure 5.1). Maintaining overview and facilitating communications in this process is one of the aims of the method presented here.

Another block diagram can be drawn that contains all life-cycle functions (see for instance [INCOSE SEH Working Group, 2000]). A diagram like that, possibly based on a story, can be used as a top-level description of all activities to be performed by the system, and all activities needed to put the system into service. When applying systems engineering to its full extent, such a top-level functional block diagram may prove to be of great value.

5.3.2 Story Telling

In [Muller, 2004b; Wojtkowski and Wojtkowski, 2002] the usage of *story telling* in systems engineering is treated. This is a textual description of the use of the system to be designed. As an example the following story can be given on the use of a coffee cup:

> On his busy day at work, Piedro needs his first cup of coffee. So he picks up his cup from the desk and heads towards the coffee machine. On his way he sees that the cup is not entirely clean. It has been a while since he took it home to give it a good wash and there is no way to properly wash up his cup at the office.
>
> Well, it is not that dirty that he cannot drink from the cup, so when he arrives at the coffee machine he places his cup in the appropriate opening, presses the series of buttons (there is no coin needed as the company provides for the coffee) and gets his extra strong coffee, with only little sugar, and no milk. Unfortunately the cup was not placed exactly right so some coffee is spilled. Making the handle dirty. Now he burns his fingers because he has to hold the cup itself, not the handle.
>
> The coffee is too hot to drink right off the machine, so Piedro walks back to his office, puts the cup on the desk and resumes his work. When he thinks about his coffee again, it is a bit cold but Piedro drinks it anyway. He needs the caffeine!
>
> On Friday when he has left the building, Piedro remembers that again, he forgot to take his cup home to give it a good wash. Then he smiles because he realizes that this way he cannot forget it to take it back to the office on Monday morning.

This story has been written only thinking about the use of the coffee cup and context knowledge about working at an office. The form of a story provides for a rich and easy to interpret description that makes discussions among colleagues easy. Before a story is analysed it should have been discussed among the designers, and whenever possible with the end-user and other stakeholders. Discussions with the latter two are possible because the use of jargon should be kept to a minimum.

When analysing the story many functions can be seen in the usage of the cup and its direct surroundings:

- signal the need for coffee/caffeine;
- move to and from the coffee machine;
- clean the cup;
- take the cup home to give it a wash;
- position the cup in the coffee machine;

- order the type of coffee;
- contain the coffee
- isolate the coffee from the hand/fingers;
- signal the temperature;
- drink the coffee;
- take the cup to work.

Please note the formulation of these functions. In each case a verb is used together with a substantive. Sometimes an extra phrase is needed to define the context. However, there is no information given about the performance (like 'accurately', 'quickly', 'hot' etc.).

Another story about the coffee cup describes its entire life-cycle, from its design to its removal. In that case the functions that relate to the production, distribution, use and removal can be found. For a coffee cup this will be relatively simple. For more complex systems like cars, aircraft or wafer steppers, this can be a complicated story. In particular the production will pose many problems that can be found early in the design process by creating and analysing these stories. Thus for each product several stories can and should be written.

From a story as given above, software engineering tends to derive *use cases*. These are short descriptions of the usage of the software system. In [Daniels and Bahill, 2004], the application of use cases as a complement to the "shall"-type of requirements common in systems engineering, is treated.

5.3.3 State-transition diagram

Story telling or scenario's are used to model the desired behaviour of a system in the user's language and from the user's viewpoint. These scenario's have to be analysed in order to be translated into a more formal form that describes the behaviour of the system. If no scenario's have been written, the behaviour of the system has to be modelled from scratch. An often used tool is the *state-transition diagram* [INCOSE SEH Working Group, 2000, p.156,182]. In such a diagram as shown in Figure 5.2, the different modes of operation are modelled as boxes, the arrows between the boxes indicate what transitions are possible between the modes. The conditions that activate a given transition label the arrows.

If we use the example of drinking coffee at the office described in Section 5.3.2, we can create a state-transition diagram for the coffee machine: Figure 5.2.

The diagram in Figure 5.2 can be seen as the top-level state-transition diagram. Most of the blocks have to be worked out further to describe the system sufficiently. Such a detail can be done in another state-transition diagram, or in pseudo-code. The latter has the advantage that it can serve as basis for the software code that has to be generated eventually.

A state-transition diagram and a functional block diagram are often mixed. This will lead to mistakes and misinterpretations. A functional block diagram can best be used when the logical sequence of events is fixed, and even continuous. A state-transition diagram can best be used when there are several (even many) different paths that describe the system's behaviour.

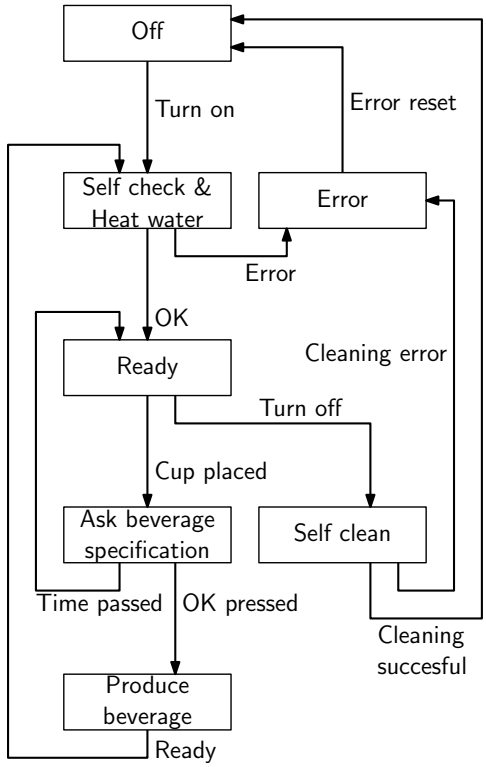


Figure 5.2: State-transition diagram for a coffee machine.

5.3.4 Influence diagram

A less well-known way of investigating the operation of a system is by creating an *influence diagram* as shown in Figure 2.9. Such a diagram consists of a pictorial representation of the system under consideration (like an iconic diagram, a sketch, block diagram) with all effects influencing the system indicated with text and/or symbols. The influences in Figure 2.9 are indicated with either ellipses or clouds. An ellipse is used when the place of influence can be clearly indicated, a cloud in other cases.

5.3.5 Mindmap

Mindmapping is a technique to order and capture thoughts. Introduced by Tony Buzan in the 1960's it has become widespread. In particular during the last years, mindmapping has achieved a lot of attention. Described in several books and documents (for instance [Buzan and Buzan, 1995] and [Gelb, 1999]*), it can be used both as note-taking and for a variation on brainstorming.

*translated from [Gelb, 1998]

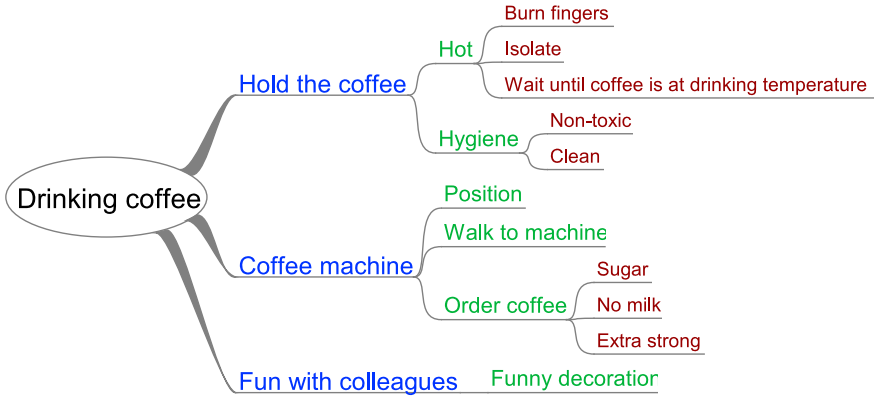


Figure 5.3: Simple mindmap of drinking coffee.

In this research, it is proposed to use mind-mapping for analysing the functions of a system. This can be done instead of, parallel to, or after the story has been created.

The easiest way to use a mindmap to investigate the functions of a product is by using it as a tool for brainstorming alone or in a team. Central question (starting point for the mindmap) should be (in the case of a coffee cup): what do I expect from a coffee cup. Using the form of a mindmap automatically orders the functions found. In general the hierarchy that follows from the first versions of the mindmap is not the best function distribution, but it can be used as a starting point.

5.3.6 Utilities

A special category of functions are what we would like to call the *utilities*. These are functions that provide service to other functions in the system. They are normally not found directly from any of the models above. Therefore special attention has to be given to finding these. Examples are:

- power generation, conditioning and/or distribution;
- memory allocation;
- climate control;
- mechanical support.

As these functions will be used by many, if not all, subsystems and they normally do not contribute directly to the key drivers they require special attention. Therefore these functions have to be separated from the other functions, as distributing these functions can be hard or even impossible, as they provide the *infrastructure* for the system under design.

5.4 Performance View on a System

Another view on the system concerns performance. Any system has to meet several performance requirements. However, the question is how the different functions in the system relate to these performance requirements. One way of distributing these is by using *system*

budgets [Frericks *et al.*, 2006; Muller, 2005b]. In the wafer stage example, the system requirements can be among others:

1. Throughput of x wafers/hour;
2. Overlay of y nm;
3. Critical dimension of z nm.

The requirements should represent the *key drivers* [Heemels *et al.*, 2006; Muller, 2004b] for the customer (the customer can be the end-user, but also the company ordering a system; see [Eekels and Poelman, 1995; Eger *et al.*, 2004a]). These key drivers are generalised requirements like *image quality* for a medical imaging system, *overlay* and *throughput* for a wafer scanner and *cost per passenger per mile* for an airplane.

The key drivers mostly do not relate to the utilities mentioned above. Of course there will be requirements specified for these utilities, e.g. maximum power consumption.

As an intermezzo, we can find the key drivers for the present research. As described in the Introduction (Chapter 1), the goal is to develop a tool that helps the system designer in gaining insight, in creating and maintaining overview, and in stimulating innovation. These goals may thus be considered as the key drivers of the research. The first two will be covered in this chapter. The latter will only be mentioned in this chapter, but will receive attention in the next.

5.5 Integrating the Functional and Performance Views

As from a system architect's view both the functions and the requirements have to be divided over the constituting parts, it appears an integrated solution may be profitable. [McDavid, 2005] advocates an architectural approach:

> "This allows specialists to focus on their parts of the problem, and to meet at well-defined interfaces, but to understand the other levels of the problem that provide a context for their work."

Specialists need to focus, but there is a serious danger that if they are not aware of the context, the system concept and system architecture, they will provide a well functioning part that does not fit in the system. Specialists may keep on improving their part, although the performance is already adequate. Or they insist they have the hardest problem to solve and try to move performance demands to other subsystems. Therefore the architect has the responsibility of providing the specialists with up to date context and system information. The other way round, the specialist has to inform the architect on his progress and results, definitive or preliminary, see Figure 4.2 on page 43.

As a tool for this process, we propose to use a scheme as shown in Table 5.1. The scheme can be used in the following manner (see Figure 5.4):

1. Identify the functions of the system on system level (Section 5.3);
2. Identify the key drivers and performance requirements on system level (Section 5.4);
3. Create a table with the functions as rows and the key drivers as columns.
4. Check every cell whether the function contributes to the key driver.
5. Create architectures by naming subsystems and assign functions to subsystems.
6. Create system budgets.
7. Repeat for next hierarchical level.

Table 5.1: Scheme that can be used to couple functions to key drivers and/or requirements. This scheme constitutes the coupling matrix $\mathcal{C}$ *(kd_j stands for a key driver.)*

	kd_1	kd_2	...	kd_m
function f_1			×	×
function f_2	×	×		
...		×		
function f_n	×	×		×

When the requirements develop, the key drivers can be replaced by system requirements and subsystem requirements.

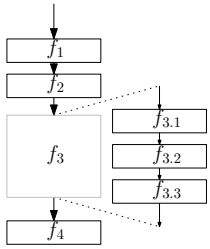
In the early phase of the design project, the scheme will be filled with crosses or ones when there is a contribution. This represents an $n \times m$ *coupling matrix* $\mathcal{C}$ that connects the functions to key drivers/requirements. This matrix can then be used to devise a basis for a system budget that distributes the requirements. In the next phase numbers can be used to investigate the distribution. One approach is to use symmetrical triangular fuzzy numbers (STFNs) [de Andres Sanchez and Terceno Gomez, 2003; Chan and Wu, 2005]. These STFNs can result from interviews with experts and previous experience with comparable products. Also, analyses of physical behaviour and/or experiments can be used to investigate individual contributions; see Table 3.4 on page 32 and [Muller, 2005b].

With STFNs vague answers of experts can be expressed as a pair of numbers that show the interval containing the expected value. The membership function has a symmetrical triangular shape. If, for instance, on a 1–6 scale, the expert thinks the actual score will be "about 2", the STFN may be [1,3], or when he is more confident [1.5,2.5]. These STFNs follow basic arithmetic rules as described in Appendix B. Addition, subtraction, and multiplication are possible, see Table 5.2, thus permitting the use in a matrix as shown in Table 5.1.

When the actual system budgets are created, and are used as basis for the subsystem specifications, the STFNs will have to be replaced with fixed numbers. This requires analysis by the system architect. When the number lies outside the interval given by the STFN, feasibility will have to be investigated thoroughly.

Table 5.2: Basic arithmetic of symmetrical triangular fuzzy numbers (see Appendix B).

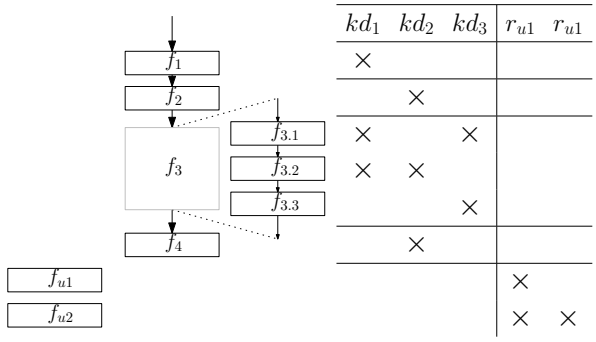
	arithmetic
Addition:	$[a_L,a_R] + [b_L,b_R] = [a_L + b_L, a_R + b_R]$
Subtraction:	$[a_L,a_R] - [b_L,b_R] = [a_L - b_R, a_R - b_L]$
Scalar multiplication:	$k \times [a_L,a_R] = \begin{cases} [ka_L, ka_R] & \text{if } k \geq 0 \\ [ka_R, ka_L] & \text{if } k < 0 \end{cases}$
RMS addition:	$\sqrt{[a_L,a_R]^2 + [b_L,b_R]^2} \approx$ $\approx \left[\frac{a_L(a_L+a_R)+b_L(b_L+b_R)}{\sqrt{(a_L+a_R)^2+(b_L+b_R)^2}}, \frac{a_R(a_L+a_R)+b_R(b_L+b_R)}{\sqrt{(a_L+a_R)^2+(b_L+b_R)^2}}\right]$



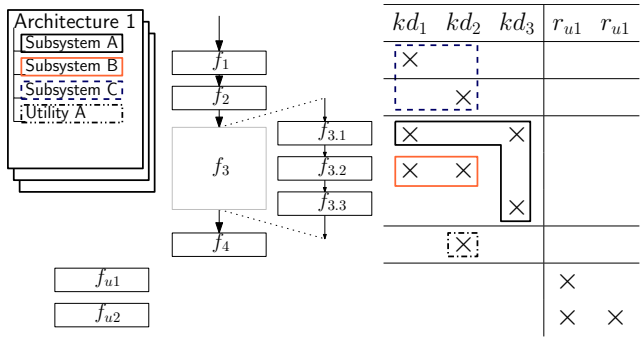
(a) Investigate functions

kd_1	kd_2	kd_3	r_{u1}	r_{u1}

(b) Investigate key drivers



(c) Create and fill the coupling matrix $\mathcal{C}$



(d) Create subsystems and architectures

Figure 5.4: The architecting tool in steps.

When this type of quantitative data is available, the matrix elements can be used to represent the contributions, either in absolute numbers, percentages, or graphical. This provides a clear and communicative representation of the system that can be kept up-to-date throughout the design process, as shown in Figure 4.2 on page 43.

In this process, columns have to be added for the requirements regarding the utilities (Section 5.3.6). In the early phase these columns may be ignored; later on they will require attention. The coupling matrix $\mathcal{C}$ can be partitioned so that the bottom right corner deals with the utilities, as shown in Figure 5.4(d).

The difference between this process and Axiomatic design is interesting. The latter states (see Section 5.2) in the first axiom to maximize the independence between functional requirements. This is often stated as maximize *decoupling*. In the approach presented above, we investigate the *coupling* between functions and key drivers/requirements. Does this mean we contradict [Suh, 1990]? No, the decoupling in Axiomatic design relates to the functional requirements and the way these are satisfied by the design parameters of the chosen implementation. In the approach presented here, we investigate inherent coupling between what the system has to do (the functions), and how that contributes to the stakeholder's satisfaction (the key drivers). This is initially without an implementation in mind. It is interesting to research a possible connection between Axiomatic design and the present approach.

5.5.1 The Architect's Playground

The process presented above is, at first, mainly analytical and inventorying. Once, on a given hierarchical level, the scheme is fairly complete, the actual creation of architectures can start by assigning functions to created subsystems. First, the architect creates an architecture by giving it a name. Then, a subsystem is created or selected from a list. This list will contain all subsystems created in this architecture, and can be extended with new subsystems. On the top level the list includes the user, the environment and the super system. One or more functions are then assigned to the subsystems. Each function has to be allocated to a subsystem. When a function is allocated, it is added to the budget for all key drivers/requirements the coupling matrix $\mathcal{C}$ connects the function to. If needed, the architect can zoom in on a function and allocate the subfunctions. That may result in an architecture with less interfaces and simpler budgets. Additionally, a rough estimate of the interfaces can be made based on the coupling matrix $\mathcal{C}$, the functional schemes created and the partitioning of the system functions. This has to be repeated on all hierarchical levels.

As creation of architectures is made easier, creating and comparing them is facilitated. The created architectures can be compared on the number of interfaces and number of required budgets. Also the principles of axiomatic design can be used to assess the architecture. This supports the architect in his search for different and better architectures. Thus the name *the architect's playground*.

In Sections 5.5.3 and 5.5.4 this is elaborated. Figure 5.4(d) shows a possible screen for this task.

5.5.2 Formal approach

To describe this more formal, first define the functions and requirements/key drivers:

Function : f_i with $i = 1 \ldots n$

System requirement : r_j with $j = 1 \dots m$

Part of r_j allocated to function f_i : $r_j(f_i)$

Then, system requirement r_j has to be divided over all functions. The fraction of r_j that is allocated to function f_i is:

$$r_j(f_i) = c_{ij} r_j \tag{5-1}$$

And the budget $\mathcal{B}_j$ for system requirement r_j is defined by the coefficients c_{ij} (with $0 \leq c_{ij} \leq 1$). Where in case of systematic budget contributions:

$$\mathcal{B}_j = \sum_{p=1}^{n} r_j(f_p) = \sum_{p=1}^{n} c_{pj} r_j \tag{5-2}$$

In case of stochastic budget contributions, equation 5-2 would produce a too strict a budget. For independent stochastic contributions root-mean-square (RMS) summation can be used. Thus if items $1 \dots z$ are systematic and $z+1 \dots n$ are independently stochastic the budget is defined as:

$$\mathcal{B}_j = \sum_{p=1}^{z} c_{pj} r_j + \sqrt{\sum_{p=z+1}^{n} (c_{pj} r_j)^2} \tag{5-3}$$

More complex budgets can be defined similarly.

It is obvious that a budget that meets the requirements satisfies:

$$\mathcal{B}_j \leq r_j \tag{5-4}$$

Using STFNs, one can estimate the feasibility and the effort required to create a matching budget in case Equation 5-4 is not satisfied. If simple numbers are used to build the budget, any discrepancy is clearly visible. Directed and appropriate actions can then be undertaken.

The coefficients c_{ij} are (derived from) the contents of the cells in the scheme in Table 5.1. An initial order of magnitude estimate for the coefficients for requirement r_j can be calculated by inverting the number of non-zero elements for a given requirement r_j. More accurate approaches to finding these coefficients use:

- information of previous products;
- information from interviews with experts in the form of actual numbers or STFNs;
- the physics related to the functions and requirements by analysing the constituting equations; or
- performing experiments.

The other way around, all requirements for function f_i can be found by listing:

$$r_j(f_i) = c_{ij} r_j \ \forall \ c_{ij} \neq 0 \tag{5-5}$$

Once the coefficients c_{ij} are estimated or determined, other representations of budgets can be created as well. As most functions consist of several (layers of) subfunctions, representing the system hierarchy, a tree like scheme shown in Figure 5.5 is a very communicative means of visualising the budget. ([Figure 7.6 in Muller, 2004b] shows an overlay budget in the form of a tree.) It also represents the hierarchy in the system design. These schemes can be used to discuss the draft budget with experts and contributors and provide context information for the specialists and detail designers.

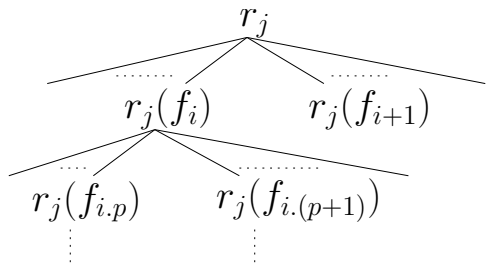


Figure 5.5: Set-up of a budget tree.

5.5.3 Implementation

Figure 5.4(d) shows a possible set-up of the screen when this proposal is implemented in a computer program. It is fairly easy to make both the functions and the key drivers/requirements zoomable: Clicking on a function expands it into its subfunctions with the corresponding block in the functional block diagram expanded, and clicking on a key driver expands it into the contributing requirements. If instead of a functional block diagram, for instance a mind-map has been used to investigate the functions, it can be shown to the left of the coupling matrix $\mathcal{C}$. Also, creation of several representations of the system's architecture and budgets is simplified. For instance, generation of the aforementioned budget trees can be automated.

This provides the system designer with overview as the design project progresses, the detail designer with context information. As the detail designer creates his design, the expected performance of the part can be input in the tool. The system designer then has an easy to use tracking tool where he can see whether the system performance is at risk. On the other hand, when problems occur at system level, the system designer has a tool to investigate the impact and find directions for solutions.

The process can base on previous designs, or can be performed for entirely new designs. When reuse of older information is required, the terms to use in the new project have to match those in the old project. This can be achieved in different ways:

- "Autocomplete": while typing a word, it is looked up in a database of previously entered terms. If a match is found, it is shown. The user can accept the term, or can continue typing.
- After a term is edited, it is analysed and terms closely related are shown. The user can choose an already used term, or maintain his entry. Other language processing tools may be suited, as well. See for instance: `http://www.visualthesaurus.com/`.
- The terms can be selected using layer menus, e.g. performance → speed → maximum speed.

The system engineers involved in the project already have to define terms unambiguously early in the design process or reuse definitions from previous comparable projects, in order to reduce miscommunication.

Table 5.3: Connecting functions and key drivers for the wafer scanner case. (CD stands for critical dimension.)

Function	throughput	overlay	CD	...
Load wafer	×			
Pre-align wafer	×			
Load wafer to exposure table	×			×
Align wafer	×	×	×	
Expose wafer	×		×	
Maintain focus			×	×
Position stage	×	×	×	
Unload wafer	×			

5.5.4 EXAMPLES

Two examples will be treated in this section. The first example, the wafer scanner, is based on previous work experience of the author at ASML (`www.asml.com`). It is a qualitative example showing the possibility of finding innovative solutions. The second example looks at a fictitious Personal Urban Transporter.

Wafer scanner

As example, the scheme in Table 5.1 is filled out in Table 5.3 for the top level wafer scanner case, see Figure 5.1. One can easily see that most functions influence throughput. Based on that scheme, one can conclude that to improve throughput, the system architecture has to be modified so that several functions do not contribute to the throughput key driver any more. In Chapter 6 we will see how this can be realised.

This example shows that the presented method is more than an analytical tool. Next to organising and bookkeeping, the method stimulates the designer to find creative solutions to the complex design problems. In particular TRIZ may complement the presented method. This if further elaborated in Chapter 6.

Personal Urban Transporter

A personal urban transporter (PUT) is a simple vehicle to be used for commuting. It should be safe, economical and should reduce the environmental load compared to the use of cars. As this is an example, not all aspects can be treated into detail.

The key drivers for such a transporter are cost per kilometre; environmental load; safety, and convenience. These can be subdivided into requirements.

- cost per kilometre:
  - cost of vehicle;
  - energy price and consumption/power.
- environmental load:
  - energy consumption/power;
  - energy and material used for production;
  - energy used when recycling.

Table 5.4: Function-Key driver scheme for the top-level Personal Urban Transporter. The columns marked Arch1, 2 and 3 show three different top-level architectures. Env: environment, U: user, PUT: Personal Urban Transporter.

Function	cost/km	env.load	safety	convenience	Arch1	Arch2	Arch3
drive vehicle							
– deliver force	×	×		×	U	PUT	U
– transmit force	×	×			PUT	PUT	PUT
maintain balance	×		×	×	U	PUT	PUT
maintain posture	×		×	×	PUT	U/PUT	PUT
create light							
– on road	×		×	×	PUT	Env	PUT
– on surroundings	×		×		Env	Env	Env
– to other road users			×		PUT	PUT	PUT
steer	×		×	×	U	U/PUT	U
navigate	×		×	×	U	U	PUT
support user	×			×	PUT	U/PUT	PUT
provide energy	×	×			U	PUT	PUT

- safety:
  - visibility;
  - protection.
- convenience:
  - convenience of support;
  - effort by user.

If we inventory the functions of a PUT, we arrive at:

- provide energy;
- drive vehicle;
- maintain balance (perpendicular to direction of movement);
- maintain posture (in direction of movement);
- create light;
- steer;
- navigate, and
- support user.

As we will see later, most of these functions consist of subfunctions.

Next step is to look at which functions contribute to which key drivers. For instance, does the function "transmit force" contributes to the key driver "safety"? No, there is no direct link between the two. If we work through all the function–key driver combinations, the scheme in Table 5.4 results. Then each (sub)function is assigned to either the PUT, the user, or the environment, in a session like the one shown in Figure 5.4(d). Table 5.4 also shows three different architectures in the rightmost columns. Arch1 shows the classic or recumbent bicycle. Arch2 represents a vehicle like the Segway (`www.segway.com`). Arch3 is a new device that will be analysed further.

Looking at the key drivers, we can define requirements for cost of the vehicle and power consumption that are part of the key driver cost per kilometre. Visibility is a part

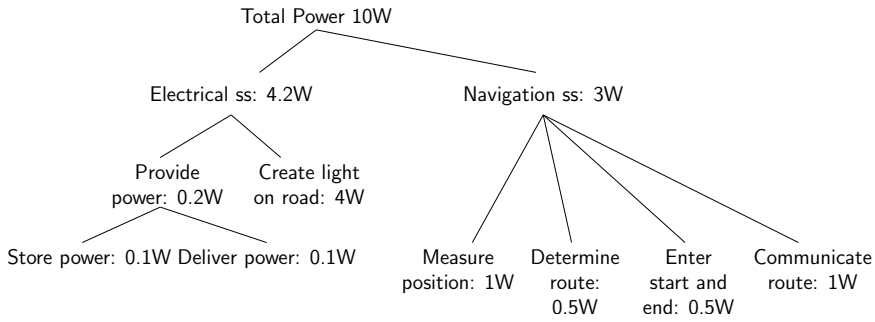


Figure 5.6: Power budget tree for the Personal Urban Transporter.

of the key driver safety and so is effort by the user a part of the key driver convenience. More requirements are needed in an actual case, but space prohibits analysing this into more detail. One of the next steps can be finding the numbers for these requirements. However, the method presented here does not rely on them. To create an architecture and corresponding budget, it suffices to indicate the *relative* part of the total requirement that is allocated to a specific function. Therefore Table 5.5 does not use actual numbers, but shows percentages of the total requirement. This way the relative importance of a function with respect to satisfying the stakeholder's interest is shown. The distribution determines the final product to a large extent. Therefore several of these schemes should be produced, and discussed with the stakeholders. We can see that creating visibility is obtained by creating light on the road, by an electrical subsystem; and by creating light to other road users, by the drive and frame subsystem. This thus poses a requirement on the designers of the drive and frame subsystem to use very visible materials, or add reflecting strips to the frame, for instance.

The power budget, can be converted into Watts, and presented in the form of a tree, like the one in Figure 5.5. The result is shown in Figure 5.6. It is easily seen that there is some margin in the budget. This could have been added explicitly as an extra budget item on the first level.

It can be seen that the budget for cost of the vehicle will not meet the requirement in the concept in Table 5.5. Next step, thus, is to reallocate the numbers, omit one or two subsystems (like the navigation subsystem) or accept the fact that the PUT will be more expensive than desired.

One of the advantages of the present method is that the kind of problems illustrated by the cost of the PUT, will surface early in the design process clearly and communicably. Thus appropriate actions can be taken in the concept phase, not as a repair action when the product is nearly finished. Another is that by creating these kind of schemes on all hierarchical levels, overview and consistency can be maintained, and context information is present for all designers.

5.6 Conclusions

A system's architecture should expand beyond the system itself. As in most cases the benefit of a system is only achieved by cooperation of the system, the user and the environment,

Table 5.5: Function-Key driver scheme of the Personal Urban Transporter with more detail and an allocation of functions to subsystems. The column headings mean: cost –cost of the vehicle, power –power consumption, visibility –how visibility is achieved, and effort –the effort by the user. The numbers indicate percentages of the requirement. The ∗ means that this is not part of the PUT-budget – the user has to deliver the force.

Function	cost	power	visibility	effort	subsystem
drive vehicle					
– deliver force				∗	
– transmit force	20				Drive and frame
maintain balance	20				Drive and frame
maintain posture	20				Drive and frame
create light					
– on road	5	40	60	1	Electrical
– to other road users	5		40		Drive and frame
steer				∗	
navigate					Navigation
– measure position	5	10			
– determine route	5	5			
– communicate route to user	5	10			
– enter start and end	5	5		5	
support user					Seat
– support user	20				
– adapt to user	5			5	
– allow movement	10			2	
provide power					Electrical
– store power	5	1			
– deliver power	5	1			
total	135	72	100	13	

both the user and the environment should be considered in creating the architecture. Functions can be assigned to the user, the environment, or the super-system.

Based on analyses of models used by designers in the previous chapter, the functional view and the performance view on a system, a method is presented that couples functions to key drivers and/or requirements using a coupling matrix. This matrix forms the basis for creation of system architectures and system budgets. As the presented method can be readily implemented in software, it will be easy to document the architecting process. Moreover, the method can be linked to TRIZ and can be used to solve problems early in the conceptual phase.

From the examples it can be concluded that the method provides overview for the system architect and context information for the designer. Architectures can be created easily and budgets for the system requirements can be deduced directly. Even without the actual requirement numbers, architectures can be created and compared by distributing the requirements relatively (in percentages or fractions of the total).

Chapter VI

TRIZ and Systems Architecting

TRIZ has gained interest over the past decades. TRIZ is widely used and acknowledged for dealing with technical issues on the component level. However, decisions on system level have a much greater impact than those on component level. TRIZ is not yet widely applied there, though. Therefore it is worthwhile to investigate applying TRIZ early in the design process.
The chapter explores the benefits of and possibilities for applying TRIZ in the architecting phase. We will base the use of TRIZ on the method presented in Chapter 5. The approach provides leads for integrating TRIZ in the system architecting phase. These will be discussed in detail as the main subject of the chapter, including examples and conclusions.

6.1 Introduction

The strategy of connecting TRIZ to the FunKey method, presented in the previous chapter, is shown in Figure 6.1. This figure shows the general TRIZ approach of generalising the problem and finding a generalised solution (also see Figure 2.8, page 22) and the contributions that will be presented in this chapter. The first step: coming from a specific problem, as described by the architecting method, to a generalised problem that can be solved by TRIZ, is what we will concentrate on in Section 6.2. Section 6.3 contains examples of application of the method. Section 6.4 draws conclusions.

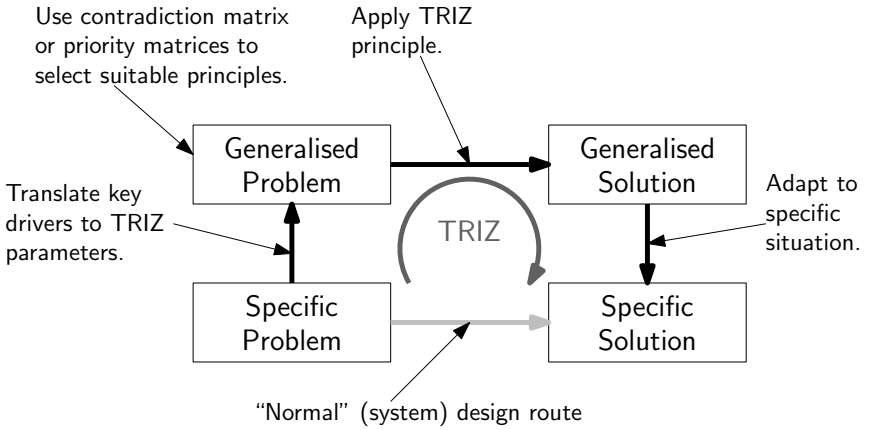


Figure 6.1: Overview of the approach to be presented. The main contribution is in the first step: generalising the specific problem and selecting suitable TRIZ principles, based on the system architecting information captured with the coupling matrix C.

This chapter is a reworked version of [Bonnema, 2006b].

We have proposed in Chapter 5 the FunKey method for system architecting. This method uses a *coupling matrix* $\mathcal{C}$ to connect functions to key drivers. In the Introduction functions have been defined (see page 5). They are well known in the TRIZ approach. They are tasks to be performed by the system like *expose wafer*, *transport sand*, and *create image*.

Key drivers are generalised requirements that express the customers' interest [Muller, 2004b]. Where it should be noted that the customer can be the end-user or the company downstream in the supply-chain. Examples of key drivers are *image quality* for a medical imaging device, *load capacity* and *cost per ton per kilometre* for a truck.

After the initial matrix $\mathcal{C}$ has been filled with crosses or ones (when there is a contribution from the corresponding function to the key driver), we proposed to quantify the contributions using either numbers, or symmetrical triangular fuzzy numbers (STFN) (see Appendix B). To facilitate the coupling to TRIZ in an early stage, the crosses or ones can be replaced by +es or –es to indicate useful or harmful contributions, respectively. This will be used in section 6.2.

The FunKey procedure visualises implicit architectural decisions. Therefore, it is a valuable tool for a team of architects and for communicating architectural decisions between architect and specialist and/or detail designer.

6.2 Connecting TRIZ to FunKey

The information in the FunKey matrix provides information on relations between functions and performance of the entire system. The following strategies will be presented that couple the system information to TRIZ:

1. Using the priority matrix by [Ivashkov and Souchkov, 2004];
2. Using useful and harmful contributions; and
3. Using insufficient and excessive contributions.

Each strategy will be elaborated in separate subsections.

For all strategies the key drivers have to be related to the 39 parameters of a technical system, as defined in [Altshuller, 1997], see Table 6.1. These are generalised properties of a system. The key drivers in the FunKey matrix are generalised requirements. Therefore a match between the two seems feasible. Problem is that the 39 TRIZ parameters are well established and fixed. Key drivers will be redefined for every new project. Some reuse might occur, but the method may not rely upon that. Two solutions can be seen for relating the key drivers (and subsequent requirements) and the TRIZ parameters: technology and experience. As for technology, one can think of using Artificial Intelligence to recognise one or more TRIZ parameters corresponding to a key driver. This kind of technology is available; see for instance `http://www.visualthesaurus.com/`.

On the other hand, as all TRIZ practitioners know, contradictions are not defined in terms of the 39 parameters from the outset. It takes some experience and associative power to find the parameters that apply directly to the problem at hand. In this aspect FunKey does not differ from normal use of TRIZ. We will therefore not look into this further, but examples are provided below.

Table 6.1: The 39 parameters of a technical system as presented in for instance [Altshuller, 1997].

#	Technical Parameter	#	Technical Parameter
1	Weight of moving object	21	Power
2	Weight of stationary object	22	Loss of energy
3	Length of moving object	23	Loss of substance
4	Length of stationary object	24	Loss of information
5	Area of moving object	25	Loss of time
6	Area of stationary object	26	Quantity of substance/the matter
7	Volume of moving object	27	Reliability
8	Volume of stationary object	28	Measurement accuracy
9	Speed	29	Manufacturing precision
10	Force (Intensity)	30	Object-affected harmful factors
11	Stress or pressure	31	Object-generated harmful factors
12	Shape	32	Ease of manufacture
13	Stability of the object's composition	33	Ease of operation
14	Strength	34	Ease of repair
15	Duration of action of moving object	35	Adaptability or versatility
16	Duration of action of stationary object	36	Device complexity
17	Temperature	37	Difficulty of detecting and measuring
18	Illumination intensity	38	Extent of automation
19	Use of energy by moving object	39	Productivity
20	Use of energy by stationary object		

6.2.1 Using the priority matrix

[Ivashkov and Souchkov, 2004] introduced an interesting use of TRIZ in early stages of design, when the amount of available analytical tools is limited. It bases on the well-known contradiction matrix that shows which of the 40 innovative principles (see Appendix C.4) has the largest chance of success for a problem. In TRIZ, a problem is often described as a contradiction between two technical parameters: improving one parameter causes the other to deteriorate. Table 6.2 shows a segment of the contradiction matrix.

From the contradiction matrix, [Ivashkov and Souchkov, 2004] derive a *priority matrix*. The number of times a given innovative principle IP_k is proposed to improve parameter p_i is determined by counting the number of times the principle IP_k is mentioned on the row for parameter p_i. This yields the score s_{ik} that shows the relevance of innovative principle

Table 6.2: Part of the contradiction matrix used in TRIZ [Altshuller, 1997]. Only the segment for the technical parameters 9–12 is shown. In the cells, the corresponding Innovative Principles are shown. The numbers refer to the principles listed in Appendix C.4.

Improving		**Worsening**			
		9	10	11	12
9	Speed		13, 28, 15, 19	6, 18, 38, 40	35, 15, 18, 34
10	Force	13, 28, 15, 12		18, 21, 11	10, 35, 40, 34
11	Tension/Pressure	6, 35, 36	36, 35, 21		35, 4, 15, 10
12	Shape	35, 15, 34, 18	35, 10, 37, 40	34, 15, 10, 14	

Table 6.3: Priority matrices in FunKey.

i. Determine which key drivers have to be improved.
ii. Use the priority matrices PM^+ and/or PM^- to identify applicable innovative principles.
iii. Apply the principles to the corresponding functions, or the system.

IP_k for improving the performance of parameter p_i. Taking the most relevant principles only in the priority matrix PM^+ thus directs the designer to the most successful principle IP_k for improving parameter p_i. This is done in Appendix C.2.

Extending this, we can define a priority matrix for worsening features: PM^-. If the aim is not to improve an already positive feature, but to minimize the impact of a worsening feature, we can create analogously to [Ivashkov and Souchkov, 2004] a matrix for worsening features: Instead of counting the number of occurrences of IP_k in a *row* of the contradiction matrix, we count the number of occurrences in the *column* of parameter p_i. The resulting matrix can be found in Appendix C.3.

These two matrices can then be used in cooperation with the FunKey approach in the following manner. After the key drivers have been coupled to functions using the coupling matrix $\mathcal{C}$, each key driver identified is connected to a TRIZ parameter. The positive (PM^+) or negative (PM^-) priority matrix is then used to select one or more promising inventive principles. Each of these principles is then applied to the functions the key driver is associated with in the coupling matrix $\mathcal{C}$, or to the entire system. A generalised solution and then a specific solution (Figure 6.1) is found as in "normal" TRIZ. See Table 6.3.

6.2.2 Using useful/harmful contributions

As mentioned in Chapter 5, the coupling matrix $\mathcal{C}$ is initially filled with crosses. Before analysing the matrix and providing numbers, one can decide to check every cross whether it is a useful (+) contribution, or a harmful (–) contribution. If an identified function f_i has a useful contribution to key driver kd_j and a harmful contribution to key driver kd_k, a contradiction can be formulated between kd_j and kd_k. Associating each key driver with a TRIZ parameter creates a reference to a cell in the contradiction matrix. The principle(s) given there can be applied to function f_i. The procedure is summarized in Table 6.4.

Table 6.4: Useful/harmful in FunKey.

i. Identify useful/harmful contributions of functions to key drivers in the FunKey matrix $\mathcal{C}$.
ii. Identify contradictions: a useful contribution to one key driver and a harmful contribution to another key driver by the same function.
iii. Use the contradiction matrix to identify applicable innovative principle(s).
iv. Apply the principle to the function.

Table 6.5: Insufficient/excessive in FunKey.

i. Identify insufficient/excessive contributions of functions to key drivers in the FunKey matrix $\mathcal{C}$.
ii. Use the priority matrices PM^+ and/or PM^- to identify applicable innovative principles.
iii. Apply the principle to the function.

6.2.3 Using insufficient/excessive contributions

Alternatively, one can examine whether a cross in the FunKey matrix corresponds to an *insufficient* or *excessive* contribution of the given function to the given key driver. These can be marked with an *i* or an *e* in the matrix, respectively. An *i* for key driver kd_j, that corresponds to TRIZ parameter p_i, directly points to the row for that parameter in the positive priority matrix PM^+. The corresponding *IP* can then be applied to the functions that have insufficient contribution to kd_j. Analogously an *e* directly points to a row in the negative priority matrix PM^-. This results in a procedure that is a mix between the procedures in Sections 6.2.1 and 6.2.2. The result can be seen in Table 6.5.

The procedures mentioned above can be implemented in a computer support program. If the FunKey matrix is created using a computer tool, the computer can suggest appropriate innovative principles to the architect.

6.3 Examples

6.3.1 Wafer scanner

A wafer scanner is the most critical part in a chip manufacturing line. The scanner images an original (called reticle) many times on the wafer. The image is reduced in size by a factor of four. One of the key drivers of a wafer scanner is *throughput*. In Table 5.3 (page 66) the

Table 6.6: Modified function-key driver scheme for the ASML TwinScan® system (see Figure 6.2). Compare with Table 5.3 and Figure 5.1.

Function	throughput	overlay	CD	...
Load wafer				
Pre-align wafer				
Load wafer to exposure table				×
Align wafer		×	×	
Expose wafer	×		×	
Maintain focus			×	×
Position stage	×	×	×	
Unload wafer				

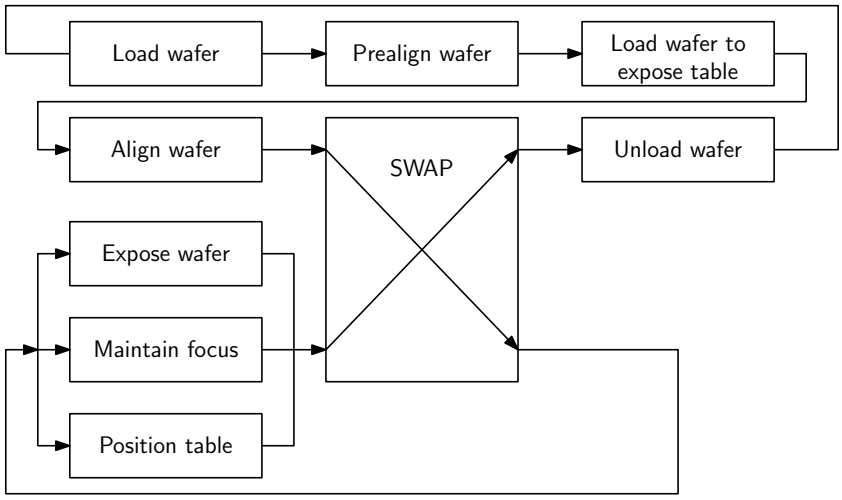


Figure 6.2: Functional block diagram of a Twin wafer scanner. SWAP swaps the two wafer tables. The one that was loading and aligning will continue as exposure table; the table that was exposing, continues as loading and aligning table.

main functions of the wafer scanner are shown, together with the key drivers and a filled out coupling matrix $\mathcal{C}$.

One can easily see that most functions influence throughput, see Table 5.3 (page 66). Based on that scheme one can conclude, as seen in Chapter 5, that to improve throughput the system architecture has to be modified so that several functions do not contribute to the throughput key driver any more.

Let us apply the procedure in Section 6.2.1 to improve throughput. Throughput relates to the TRIZ parameter 39: *productivity* (see Table 6.1). The priority matrix PM^+ (Appendix C.2) then suggests to apply innovative principle 10: *prior action*. We can perform all but expose wafer and position stage in advance. This is realised in the TwinScan® systems by using two simultaneously moving wafer tables (see Figure 6.2 and [Loopstra *et al.*, 1999]). One performs measurements, the other one exposes a wafer. This way, the serial connection in Figure 5.1 has been removed, resulting in a function-key driver scheme shown in Table 6.6.

6.3.2 Personal Urban Transporter

As second example, part of the Personal Urban Transporter (PUT) introduced in Section 5.5.4 will be analysed. A PUT is a small, safe and economical vehicle for commuting. Based on an initial analysis of the system as shown in Chapter 5, several functions have been assigned to the PUT. In Table 6.7 part of the FunKey table is filled with +es and –es.

We can associate key driver cost per km with TRIZ parameter 19: use of energy by moving object, safety with parameter 30: object affected harmful factors, and convenience with parameter 33: ease of operation. For the function maintain posture the contradiction between parameters 30 and 19 is identified, leading to innovative principle 24: mediator.

Table 6.7: FunKey for the Personal Urban Transporter (PUT).

Function	cost/km	env.load	safety	convenience
drive vehicle				
– deliver force	–	–		–
– transmit force	–	o		
maintain balance	–		+	+
maintain posture	–		+	+
create light				
– on road	–		+	+
– to other road users			+	
steer	–		+	–
navigate	–		+	+
support user	–			+
provide energy	–	–		

This leads to an airbag around the user, to be used when he is about to lose his posture (=fall over). For the contradiction between parameters 30 and 33 for the function steer, one of the TRIZ principles is 25: self-service. This leads to a steering system that uses the edge of the road to steer the PUT.

6.4 Conclusions

We have presented a way to connect TRIZ to the architecting method FunKey that was presented in the previous chapter. This may help the system architect both in finding implementations for his functions, and in simplifying the system. This accommodates for the innovation part of the goal presented in Figure 1.2 (page 8). Also, the solution found appears to be easy to implement in a computer tool. Main issue is how to connect the key drivers with the TRIZ principles. This can either be achieved with artificial intelligence, a database of related terms, or by using the experience of the designers. Latter solution is preferred as for now. As seen from the examples provided, this associating of key drivers to TRIZ parameters is easily done. The examples also showed the power of the FunKey-TRIZ combination in the system design phase. By regarding the system before designing, but purely on the functions and key drivers, the architecture of the system can be altered so that it fits the customers needs better.

Both the FunKey method and the linking to TRIZ are tested in industrial cases. Results of these cases will be treated in the next chapter.

III Application and Evaluation

Chapter VII

Application of FunKey Architecting

Systems architecting is the design phase where the top level functions of a system are distributed over the system's parts, its environment and its users. Up till now, system architects had to largely learn the required skills in practice. Some courses exist (Stevens Institute, Embedded Systems Institute) that teach the right attitude and mindset for the system architect. However, methods for architecting that can be implemented in a computer tool are virtually non-existent.
Earlier we have presented a method that may aid the system architect in the early phase of design. In combination with TRIZ a design tool is created.
Main topic of the chapter is application of the method in two industrial cases. The one case is an environment where new technology has to be developed and state of the art physics have to meet machine construction principles. The other case is in an industry where well proven technology is used in such a way that high performance machines are created. The context of each case and the results will be described. A third application of the FunKey tool is performed by students at the University of Twente. We will report on their results as well.
The chapter ends with conclusions.

7.1 Introduction

We have proposed (Chapter 5) a method for systems architecting. We showed two simple examples to illustrate the approach. These were based on previous work experience and academic reasoning. The two industrial cases that will be presented next are real situations, where the FunKey method has been applied to actual system design situations.

The FunKey procedure visualises architectural decisions formerly made implicitly. Therefore, it is a valuable tool for a team of architects and for communicating and discussing architectural decisions between architect and specialist and/or detail designer. For more information, and particularly the relation between the presented method and other methods like Axiomatic Design [Suh, 1990] and QFD [Chan and Wu, 2002, and references therein], the reader is referred to Chapter 5.

The chapter will look at the application of the method. First we will treat two practical cases. We will present a case of designing a machine at the edge of current state of the art. The second case is a mature field of technology, where the FunKey method is used to find new approaches to the problem at hand. Another application is where FunKey has been presented to students, to aid in a system design project. Results and observations thereof will be presented as well. We will conclude the chapter with a discussion on the results and conclusions.

7.2 Introduction to the Cases

The research is aimed at creating a practical method that supports the system architects. Although the method is still in development, we have decided to already apply it in practical

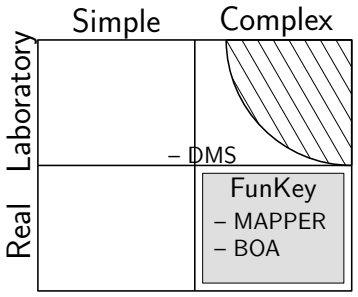


Figure 7.1: The cases of application of FunKey mapped in a space of real versus laboratory and simple versus complex.

cases in order to get early feedback on the usability, applicability and communicative values.

Figure 7.1 shows four classes of application: the applications can be Simple or Complex, and Real or in a Laboratory-setting. Where it can be noted that it is nearly impossible to devise a complex laboratory problem. To perform a comparative study, one should have a set-up where the outcome with the method can be compared to an outcome without the method, like is done in for instance [Ruder and Sobek, 2007]. For such a comparison, a laboratory setting and thus a (relatively) simple case will be used. However, as marked in Figure 7.1, that is not where FunKey is applicable. Thus a comparative set-up, though methodologically sound, does not fit the FunKey application area.

FunKey is aimed at dealing with real complex situations, as marked in Figure 7.1. Application in a laboratory environment and using a simple case would render the method useless as the problem is not complex enough to require additional tools and methods. The area marked grey is where FunKey is appropriate and where two real cases have been placed: the MAPPER case (Section 7.3) and the BOA case (Section 7.4). This, however, means that we cannot perform a comparative study and thus that there is a large asymmetry in the information available to the researcher and to others [Dorée, 1996]. We try to reduce the consequences of that disadvantage by documenting observations and experiences by the researcher and others involved. This participating approach has been used earlier to collect practical data [Finger and Dixon, 1989a, b].

In addition to the two participating cases, five student teams were instructed about the FunKey method as part of an educational project (marked DMS in Figure 7.1). This project, relatively simple for experienced designers, is complex to the third year's students. Some teams did use FunKey, others did not. Their work is presented in Section 7.5. This way some comparison is possible.

Another issue relates to the stage of development. As the method is aimed at being implemented in a computer tool, a prototype of the tool would be useful. On the other hand, when effort is put into creating a prototype tool, but the method in itself is not good, the effort is wasted. Considering the fact that no complicated nor many calculations are required, this is assumed to be no serious drawback. It only means that in addition to general purpose software (a spreadsheet program, word processor and a mind mapping program) some handwork is involved for finding the contradictions (to connect with TRIZ) and maintaining correspondence between different models. One can say that some functions that should be assigned to the tool, are now assigned to the researcher. Most important drawback is that "the architect's playground" is not a real playground. Instead, creating the architectures and subsequent creation of system budgets requires a significant amount of handwork.

The method has been directed to solving design issues where a feasible solution has to be found in a large search space. It was not intended for situations where an optimum solution has to be sought. Nevertheless, both situations have been evaluated in order to determine the general applicability.

We expect to find that the method provides overview and insight (Figure 1.2, page 8). In addition to that, using the connection with TRIZ, there might be several interesting new

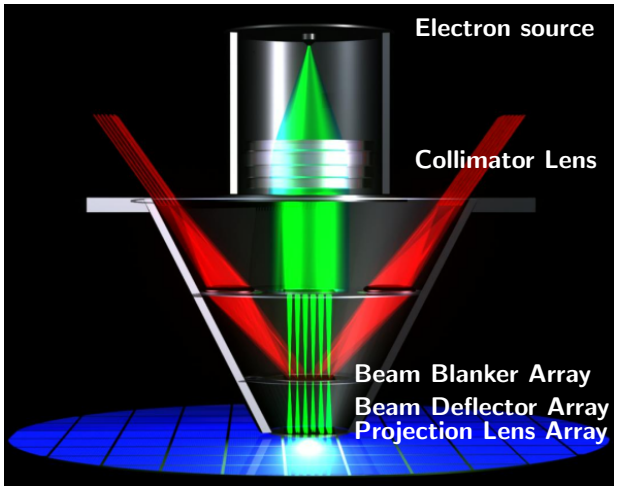


Figure 7.2: The electron optics of the MAPPER lithography system. Picture courtesy MAPPER Lithography.

ideas to be investigated further. This relates to the innovation in Figure 1.2. But we also expect the work flow to be hampered by the absence of an integrated computer tool.

7.3 Case 1: MAPPER

MAPPER Lithography (`http://www.mapperlithography.com`) is a Dutch company working on a new approach to exposing semiconductor wafers. The main differences with conventional wafer scanners are the elimination of the pricey masks (reticles) and the use of a massive amount of modulating electron beams in parallel, instead of projecting light through a mask. Some of the characteristics are the use of MEMS (Micro Electro Mechanical Systems) for the projection optics and telecom technology for transferring large amounts of data from the mask design to the electron writing system. See Figure 7.2 and [Kruit, 2007] for an overview of the electron-optical system. In this section we will concentrate on the wafer positioning system that is needed for scanning the wafer sufficiently accurately under the projection lens.

The company is currently designing a first prototype machine. This machine will be able to actually expose 300mm diameter silicon wafers in a fab-like environment to enable customers to develop their process. Additionally, the system design should accommodate for future throughput and resolution upgrades. The design process at MAPPER has been based on systems engineering guidelines as described by [Blanchard and Fabrycky, 1998; INCOSE SEH Working Group, 2000]. For instance, a clear tree-like document structure is maintained to relate specifications on different hierarchical levels to each other.

The following is an accurate report. It may be incomplete at times where information is not yet released for publication by MAPPER.

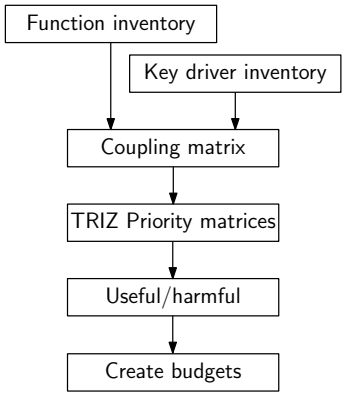


Figure 7.3: Application of FunKey to the MAPPER case.

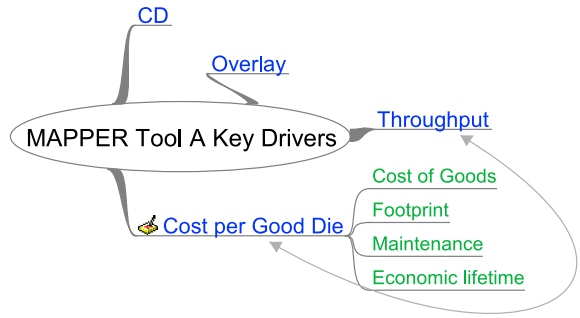


Figure 7.4: MAPPER key driver analysis. The arrow connects two related key drivers. CD = critical dimension.

7.3.1 The problem

MAPPER started from the department of applied physics of the Delft Technical University. The patented technology was developed by Prof. Kruit and his team of physicists. This guaranteed a solid physical foundation for the machine. Translating these physical principles into a machine design requires other contributions. As nanometre precision is required in a vacuum environment, many disciplines are involved.

The author, a former system engineer at the other Dutch wafer stepper manufacturer ASML (`http://www.asml.com`), has been asked to assist (part time) in the creation of a design for the wafer positioning subsystem. This involved, among others, modelling throughput and overlay. In parallel to a more traditional approach, the author has applied the FunKey method.

7.3.2 The approach

As MAPPER had several concrete issues that required investigation, this was done first to become acquainted with the company and the system. Next, the FunKey method was applied as shown in Figure 7.3.

The function inventory has been created using a mind map, see Figure 7.5. Other approaches are the use of functional block diagrams (FBD), scenario's, story telling etc. In this mind map, the hierarchy shown is not a functional hierarchy, but merely represents the author's mental model of the MAPPER exposure system. Analogously, the customer key drivers can be found in Figure 7.4. Note the similarity with the key drivers shown in the example in Chapter 5, due to the fact that in both cases, a wafer exposure system is considered. The coupling matrix $\mathcal{C}$ that is defined by the functions (rows) and key drivers (columns), is shown in Table 7.1. The spreadsheet created allows for folding columns and/or rows. This simulates zooming in on key drivers and/or functions, respectively.

The creation of the function tree, key driver tree and filling out the coupling matrix has taken approximately two man-days. The effect of this effort has been increased insight in the machine structure for both the author and colleagues at MAPPER.

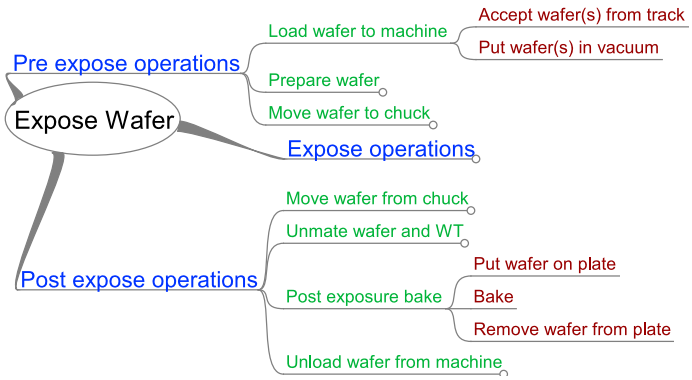


Figure 7.5: Part of the MAPPER function analysis.

7.3.3 Results

Priority Matrix

Based on the filled out coupling matrix, several ideas have been generated using TRIZ. Looking at the key drivers and their effect on the entire system, we can find TRIZ principles to improve them. For this we use the priority matrices presented in [Ivashkov and Souchkov, 2004] and Chapter 6 (see Appendix C).

CD (critical dimension, the smallest line width that can be imaged), relates to TRIZ parameter 29: Manufacturing precision. Using the positive priority matrix PM^+, principle 32 (Colour change) is suggested. This is the reason why the wavelength used in wafer steppers has changed over the years (from 365, to 248, to 192nm, to electron beam).

Overlay relates to parameter 37: Difficulty of detecting and measuring. PM^+ suggests principle 28: mechanics substitution. All measurements are now performed by optics and other contactless sensors instead of mechanical sensors. This is done to increase the accuracy (optical sensors are more accurate), to avoid mechanical vibrations from disturbing the optics, and to ensure vacuum compatibility.

Throughput relates to parameter 39: Productivity. PM^+ suggests principle 10: Prior action. In other words: do as much actions as possible prior to exposing. This is what the ASML TwinScan® machine does [Loopstra *et al.*, 1999]. However, the TwinScan® has been designed as a high-throughput tool from the outset. The MAPPER tool does not have that high emphasis on throughput.

For Cost per good die, there is no direct relation with any of the TRIZ parameters. However, the sub key drivers Footprint and Maintenance can be related to TRIZ parameters 6 (Area of stationary object) and 25 (Loss of time). Resolutions are, using PM^+, principle 18 (Mechanical vibration) and 10 (Prior action), respectively. The latter idea can be translated in reducing maintenance time by doing most adjustments and calibrations prior to mounting the parts in the machine. This is a severe demand, but essential during the integration phase, and when producing the machine in (small) series. If, namely, for adjusting or calibrating part A, part B is required, the integration cannot continue if part B is absent. Also, there is a

*Table 7.1: Function-Key driver matrix for the MAPPER case. CD: Critical Dimension, OV: Overlay, TPT: Throughput, COG: Cost of Goods, FP: Footprint, EO: Electron Optics, WT: Wafer Table. "X": Contribution, "**X**": contribution by all subfunctions, "x": possible or indirect contribution.*

Functions	Key Drivers				
				Cost per Good Die	
	CD	OV	TPT	COG	FP
Pre expose operations					
Load wafer to machine				X	
Prepare wafer				**X**	
Move wafer to chuck				**X**	X
Expose operations					
Measure EO	X	X	X	X	
Align wafer to EO		X	X	X	
Maintain focus	X	x			
Measure wafer height and tilt	X			X	
Position wafer	X	X	X	X	X
Measure wafer position	X	X		X	x
Scan wafer			X		X
Create Electron beams	X		X	X	?
Write Pattern	X	X	X		
Post expose operations					
Move wafer from chuck			X	**X**	**X**
Unclamp wafer/WT				X	
Make move				X	
Unmate wafer and WT				**X**	
Post exposure bake	X	X?		**X**	**X**
Unload wafer from machine				**X**	**X**
Release vacuum				X	X
Move wafer to track				X	X

large chance of dead-locks. This idea is not new, and can be found in many places. The use of mechanical vibration is not seen as a serious option to reduce footprint or maintenance. The second most promising principle for improving area of a stationary object is principle 2: taking out. This is a very straight forward approach: take as much functionality out of the machine and put it in the service areas of the wafer fab. This is a design decision already taken, see Figure 7.6.

Useful/Harmful contributions

Let us now look at some of the contradictions found using the FunKey matrix. If we substitute the X-es in the matrix with a + when there is a useful contribution and a – when

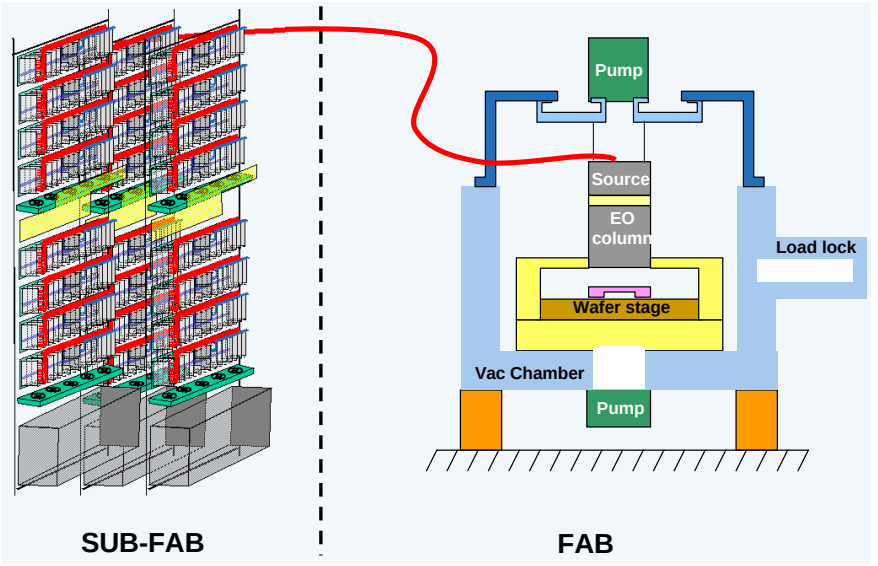


Figure 7.6: The MAPPER system, including Fab and Sub-Fab division. The Fab is the clean room where the wafers are processed. The sub-fab is a less clean area where supporting subsystems are placed. In the MAPPER case the exposure tool is placed in the fab, whereas the data system is placed in the sub-fab. Picture courtesy MAPPER Lithography.

there is a harmful contribution (see section 6.2.2), we can find, among others, the following contradictions. The subfunction "Measure EO" (not visible in Figure 7.5) has a positive contribution on CD and Overlay, but a negative one for Throughput. The Improving TRIZ parameters are measurement accuracy (28) and manufacturing precision (29). Worsening parameters are speed (9) and productivity (39). Relevant principles are found using the TRIZ contradiction matrix [Altshuller, 1997]. Application of a few of these principles results in the following: 10: Prior action: Measure the electron beams before the wafer is loaded, or while wafer load and unload is in progress. 13: 'The other way round': Do not move the chuck, but only the electron beams. 18: Mechanical vibration: Instead of moving with the chuck, let it vibrate. When the measurement system is fast enough this may work.

Other results

The architecting aid has been harder to test, as a computer implementation would provide an easier and richer user interface. Consequences on decisions can be communicated easier to the architect. An attempt has been made using the attribute manager in the mindmap program (Freemind, `http://freemind.sourceforge.net`). Using filters one can easily see all functions assigned to a given subsystem. This enables detail-designers to concentrate on their relevant information, while the system designer can hide detail issues. This results in bridging the contradiction identified in [Frericks *et al.*, 2006], that a detailed budget results in loss of overview. By interactively hiding and exposing parts of the budget both overview

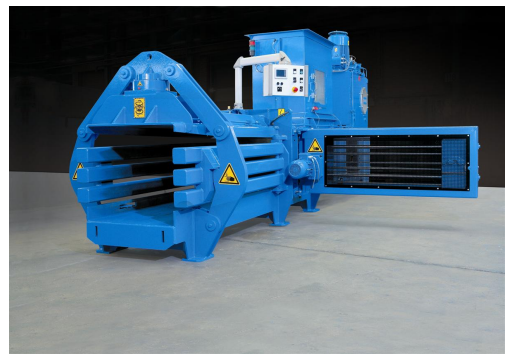


Figure 7.7: BOA Impress® and Continette® balers. Pictures courtesy BOA Recycling Equipment.

and detail can be provided, albeit not simultaneously.

In addition to the results above, an initial overlay budget has been created largely based on the coupling matrix in Table 7.1. A throughput budget had been created before. The budgets have been refined using additional calculations, modelling and interviews with key specialists at MAPPER. The budgets are not included in this thesis for competition reasons. Other budgets (for the key driver Critical Dimension and for subsidiary requirements like power consumption, footprint etc.) can be derived as well.

With this approach the architect has to make conscious decisions on each allocation and document them directly. Also alternatives are presented clearly to the architect. Apart from organizing the architect's thoughts, it informs his colleagues (both system designers and detail designers) on his reasoning. One of the colleagues at MAPPER stated: "This provides good overview over the entire system."

7.4 Case 2: BOA

BOA Recycling Equipment, with a history dating back to 1956 (`http://www.synmet.nl/`) has been producing industrial (waste) balers for over 50 years. With over 3,000 waste-disposal installations sold worldwide, BOA is widely regarded as one of the world's top players in its market. The balers (Figure 7.7) are used to compress various waste materials to reduce its volume. This way the cost of shipping the waste for further processing is reduced significantly.

Referring to Figure 7.8 a baler functions as follows. Waste is dropped in the filling funnel. The two flaps separate a volume of waste from the supply in the funnel and pre-compress that volume. Then the ram moves the waste to the left into the additional compression section (the knife cuts the waste from the supply in case no flaps are used). The ram moves back to compress another bale. The bales in the additional compression unit provide counterforce because of the taper, and are simultaneously compressed some more, when the next bale is moved to the left. The binding unit binds the bale by applying straps (metal, plastic) that avoid decompression after the bale has left the baler. Finally, the bale is ejected from the baler.

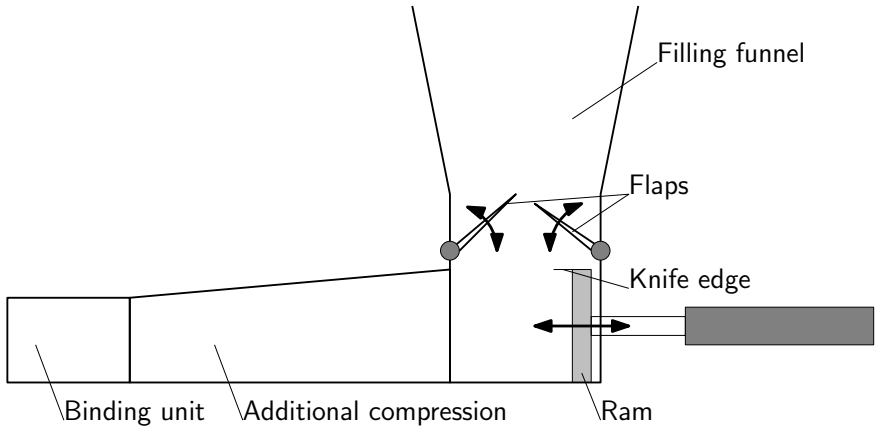


Figure 7.8: BOA waste baler system.

Through the years the design changed from a vertical to a horizontal baler. First the binding was done by hand, which was automated later on. The efficiency of material collection and compression was increased through the use of one or two pre-compression flaps as an alternative to a knife fitted on the ram.

7.4.1 The problem

At this point in time, major modifications have been implemented, yet the compression machines produced today are quite similar to 30 years ago. The balers have a capacity that depends on factors like: weight, pressing power, installed power, material size. A new design solution has to be found that surpasses current system architectures to improve the balers' capacity. If possible, the design should be modular. Furthermore entirely different ways of compressing material are to be found and developed.

7.4.2 The approach

In order to come to a conceptual design and to reach the desired design specifications and requirements, a number of inventive design methods have been used:

- (Advanced) Systematic Inventive Thinking (SIT) [Goldenberg and Mazursky, 2002];
- TRIZ; and
- the FunKey method.

In this chapter, we will concentrate on the latter in a similar manner as done with the MAPPER case (Figure 7.9). (A)SIT and TRIZ applied to a baler are treated in [Alink, 2007]. After extensive talks and interviews with engineers, a thorough insight in the operation of the balers at BOA systems was gained. Using the acquired knowl-

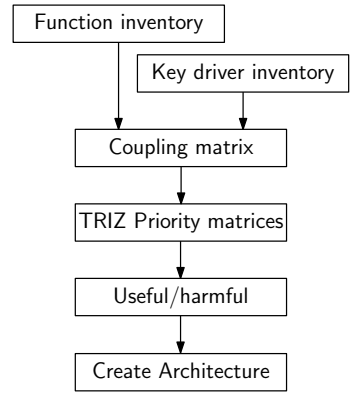


Figure 7.9: Application of FunKey to the BOA case.

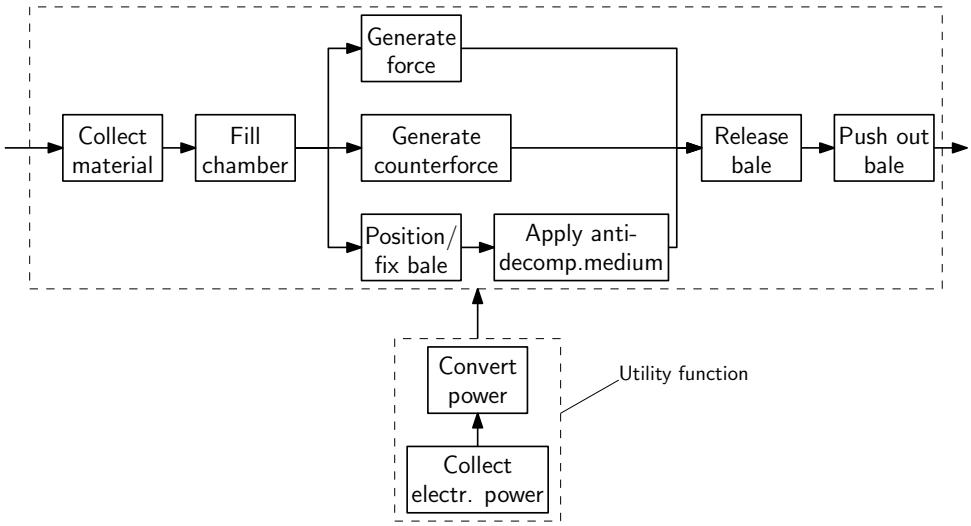


Figure 7.10: Functional block diagram of a baler

edge of the working principle the most important generalized functions were determined, see Figure 7.10.

Next step is to derive the key drivers related to the material compression machine:

- Material throughput (MT) (tons/hour);
- Volume reduction (VR), representing a number of customer interests:
  - compression ratio ($\rho_{\text{compressed}}/\rho_{\text{uncompressed}}$);
  - specific weight of bale ($\rho_{\text{compressed}}$);
  - minimal and maximal bale weight.
- Bale handling (BH), representing the following requirements:
  - easy ejection of bales;
  - bale shape;
  - bale shape consistency;
  - bale sizes (in order to transport them in sea-containers);
  - type of binding (steel, plastic, foil);
  - controllability of bale dimensions.
- Cost per ton of material processed (Cost).

Part of the resulting FunKey matrix is shown in Table 7.2, where the functions of Figure 7.10 are used.

7.4.3 Results

Priority Matrix

The key driver Material throughput relates among others to TRIZ parameters weight of moving object (1) and quantity of substance/matter (26). In both cases the suggested principle is IP 35: Transformations of properties. Several ideas have been generated. One

Table 7.2: Part of the FunKey matrix for a baler. (i: insufficient, e: excessive, +: useful, –: harmful)

		MT	VR	BH	Cost
Fill baler	Collect material	i			–
	Fill machine/chamber	i	i		–
Compress material	Generate force	+ / i	+	+	–
	Generate counter-force		+	+	–
Prevent decompression	Position/Fix bale	–			–
	Apply anti-decompression	–	i	i	–
Remove bale	Release bale	+			–
	Push bale out	+		+	–
Power conversion	Collect (electrical) power				–
	Convert Power	+ / i	+		–

is the use of a flexible compression chamber, where an external pressure is applied. This can provide for additional compression, or fine-tuning of the bale size and shape. Another suggested principle is IP 1: Segmentation. The concretization of this idea results in a ram segmented in time and/or space. Segmentation in space would mean a series of rams that each have a very limited stroke, but a large force for compression.

Useful/Harmful contributions

Although only a few harmful contributions have been identified, several contradictions can be formulated and solved. One being the contradiction between volume reduction and cost per ton, for the generate force function. Resulting inventive principles are spheroidality/curvature (IP 14), materialising into a rotating excenter ram; and thermal expansion (IP 37), materialising in compressing the waste at an elevated temperature. Cooling to room temperature will yield an additional volume reduction. The other cells in the matrix have also been investigated and many possible concepts have been identified [Alink, 2007].

Other results

Using the FunKey approach four subsystems and one utility are identified: Material entry system, Compression system, Binding system, Hydraulics system and the Structural utility system. Subsequently, all functions identified have been allocated to these subsystems. Next step is to use this for compilation of system budgets for the key drivers Material throughput, Volume reduction, Bale handling and Cost per ton and subsequently for the requirements.

In this case the FunKey method effectively dissects the system at hand. This provides valuable leads for applying TRIZ and SIT, as well as an enlarged insight.

7.5 Design of Mechatronics and Systems Project

The third year of the bachelor programme of Industrial Design Engineering at the University of Twente [Eger *et al.*, 2004b] contains a student project that deals with designing mechatronics and systems: the Design of Mechatronics and Systems project (DMS). This project is based on, and uses the same set-up as the project described in [Bonnema *et al.*, 2005], however the sensors and actuators course has been replaced by a course on mechatronic design. The basic concept of the student project is to immerse the students in a new technology, with a relatively large and multidisciplinary problem. By providing them with content information, in this case the course on mechatronic design, and procedure information, a basic course on systems engineering, the groups should be able to organize themselves. The result of the project has to be a system design and concept design of the main modules. No prototype is built, though. The groups are larger (12–15 persons) than for all other student projects of the curriculum (4–8 persons), and there is no tutor assigned to the groups. Instead, the students are encouraged to consult teachers whenever there is a need to. Note that the entire curriculum for Industrial Design Engineering at the University of Twente is project-based.

The DMS project has run now four times. Three times with the sensors and actuators course, and once in 2007, with the mechatronic design course. This was also the first year FunKey was included. In 2007 the students had to design a computer controlled table soccer game, where one or two people can play table soccer against a computer. For this, the system has to be able to control the rods of puppets on one side of the table, and detect the position, and possibly the speed and direction, of the ball. Other functions of the system can be tracking the score, provide game feedback, putting the ball into play, act as a referee etc. Also, the application environment of the soccer table has to be investigated. Will it be marketed as a gadget to consumers, or as a game to be operated in arcade halls? The five groups each had to deliver a system set-up after two weeks, and subsystem requirements and concept designs plus an integration plan at the end of the project (seven weeks after its start). At the two milestones the results are presented using posters to the teachers and peers. The teachers evaluate each poster with a written feedback and a mark on a 1–10 scale.

In the basic course on systems engineering, different systems engineering tools are treated in a practical way, see Table 7.3. The INCOSE handbook [INCOSE SEH Working Group, 2000] is treated as a tool-kit for handling large and complex projects. As an addition to the tools in the handbook, the FunKey method has been presented as a tool, in an early phase of the students' project. The connection with TRIZ as presented in Chapter 6 has not been treated, though. As the tools are treated as part of a tool-kit, there is no pressure put upon the groups to use a particular tool, or not.

Of the five groups, four have mentioned the FunKey procedure to acquire and organise information in the early phase of the project. The one group that did not report about it, may have tried using FunKey. Interestingly, though, of the four reported uses of FunKey, only one group stated they have used FunKey to create several architectures and comparing them afterwards. We will summarize the results for the system design (first milestone) of each of the five teams below. The second part of the project will be discussed only when necessary. In order to not violate privacy rules, the teams are numbered Team A to E, without correspondence to the original 1 to 5 numbering. The analysis below bases on the posters submitted and the feedback given to the groups.

Table 7.3: The System Engineering (SE) subjects treated in the SE course that is part of the Design of Mechatronics and Systems project.

1. Introduction, history, purpose and future; Why SE for industrial designers? SE in the design process, SE as repeating cycles, SE and the life cycle.
2. The role of the systems engineer in complex and simple situations; The systems engineer as "lubricant", Architecting (including FunKey architecting).
3. Lecture by guest Gerrit Muller (author of a.o. [Muller, 2004a, b]).
4. Functional analysis; Functional mapping; System budgets.
5. Planning and control; SEMS (SEMP), Monitoring progress; Documentation; Reviews; Decisions; Open items databases plus management.
6. Risks and Failure Mode and Effect Analysis (FMEA).
7. Reliability, Design against failure, (non-)redundancy; System integration and test; SE Product Control.

7.5.1 Results

Team A — The team mentioned the use of FunKey, however it appears they have only used the first stage where the functions are connected to key drivers, without allocating budgets. Whether they have used it for creating architectures is questionable. The N^2 diagram of the architecture developed can be found in Appendix D.2. The main concerns from the poster and the presentation relate to the large number of interfaces (see the N^2 diagram) and the lack of coherence between the various parts. This lack of coherence has not been eliminated in the second part of the project, where the subsystems had to be worked out further and an integration sequence had to be devised.

Team B — This team did not use FunKey. Yet they have tried to provide coherence within the project. Some crucial aspects have not been treated enough to make informed design decisions (data-interfaces in general, and the interface between the processing and feedback subsystems in particular). The second part of the project has provided not too good results.

Team C — This team has done a very good job; the best system design we had seen until now. The tools provided were used and FunKey has been the centre-point. By using FunKey to create several architectures, and comparing them using the coupling matrix and an N^2 matrix, a clear and well-defined architecture has been created. Table 7.4 shows the coupling matrix as created by this team. The N^2 diagram can be found in Appendix D.1. From the second part of the project it turned out that this clear architecture has been very helpful.

Although the game has to provide, by nature, entertainment while playing, there is only 20% of the "entertainment" budget allocated to actually moving the puppets. It is clear that a large part of the fun has to come from the fact that the table has to be computer controlled as 15% of the entertainment value comes from the visual feedback. The vandalism resistance poses a heavy burden on the design team, as can be seen from the larger than 100% budget allocated. As with the cost of the PUT in Tabel 5.5 a careful review of the requirements and

Table 7.4: FunKey matrix of Team C of the Design of Mechatronics and Systems project.

Functions	Key Drivers				
	Costs/ game	Enter- tainment	Comfort	Safety	Vandalism resistance
Turn on/off	1		5	2	10
Pay game	3		8	10	10
Set-up game					
Choose preferences	3	6	4		4
Set preferences	3	6	4		4
Feedback of preferences	2		5		5
Play game					
Insert ball	5	13	6	8	10
Detect ball	15				15
Detect puppets	15				15
Process data	10				3
Make decision	4	10			3
Actuate puppets	25	20	10	10	20
Auditive feedback	2	10	5	1	5
Visual feedback	4	15	7		5
Save data	2		5		2
Detect goal	3	3	5	5	5
Exchange data	3	10	2	1	5
Totals:	100	93	66	37	121

opportunities is needed in this area. The second part of the project has resulted in good and well-founded concept solutions, with the exception of one sub-team. All subsystems appear to be well-integrated into the system. The system engineers have monitored the overall performance of the system during the design process by using budgets.

TEAM D — This team did a proper stakeholder-analysis. Just as Team A, only the first stage of FunKey has been used. As a result, there is a lack of coherence between the activities performed. The way functions are allocated to subsystems is not clear, and even ambiguously. This lack of coherence has continued during the second part of the project, even though the team was given feedback on this issue.

TEAM E — This team has applied the FunKey method *after* already having created an allocation of functions to subsystems. It appears as if they have not done anything with the coupling matrix they have created. One of the comments by the teachers, at the first review, was that the connection between different parts and subsystems is lacking. Also the goal of this team is to create a system that meets every possible wish, while a strategy to choose

Table 7.5: Summary of the experiences and observations with FunKey in the different application cases. DMS –no FunKey refers to the DMS teams that did not use FunKey (n.a.: not applicable).

Case	Insight	Overview		Innovation
		create	maintain	
MAPPER	✓	✓	n.a.	✓
BOA	✓	✓	n.a.	✓
DMS – with FunKey	✓	✓	✓	n.a.
DMS – no FunKey	∼	∼	∼	n.a.

between conflicting interests is lacking. This has lead to poor results in the second part of the project. With a filled out coupling matrix, like the one shown in Table 7.4, the conflicts emerging from this goal would have surfaced early in the process.

7.5.2 Observations

From the above it is clear that the team that applied FunKey well (Team C), had the best overall results, both in the first and the second part of the project. This is even more relevant as their results exceeded those of all groups that took part in the DMS project and the previous version, the SAS project. 2007 was the first year FunKey was treated, and this has positively influenced the results. Although some other teams have performed well, often coherence was lacking. In particular the teams that did not use FunKey as an architecting aid (Team B, Team E) have shown a lack of coherence in their results compared to the one that did use FunKey. As the students are third year's students, they lack experience in working in large and complex technical projects. This accounts for some lack of quality and depth.

7.6 Discussion

Table 7.5 shows a summary of the experiences and observations with the different cases where FunKey has been used, respectively not used, in relation to the research goal described in Chapter 1. Although the absence of a prototype computer tool has hampered the work flow in the industrial cases, it is clear that in both the MAPPER and the BOA case several interesting concepts have been found that can be investigated further. Although the concepts may have been found in other ways, the FunKey method has concentrated the effort using the TRIZ philosophy. This shows the innovation aspect of the tool. In addition to the new concepts found to simplify the machines, insight in how the machines work has increased. The students that applied FunKey early in the project and used the results, have shown improved insight in the problem and the stakeholder's interests. It is therefore expected that this will hold for any system the FunKey method is applied to.

By creating the FunKey diagrams, overview has been created in the complex systems in the MAPPER and BOA cases. In the DMS project, the system designs of Team C that has used FunKey clearly showed it was designed with a better overview of the problem than the other system designs.

As for maintaining overview, this could not be verified in the industrial cases because of the limited time the projects have run, and the fact that the designers applying the method were not appointed system engineers. In the DMS project the infrastructure created with FunKey has helped the students that took the role of system engineers in maintaining overview, as reported by the system engineers in Team C.

In discussions with engineers the difference between function, behaviour and implementation turned up, also see [Shimomura *et al.*, 1995; Tomiyama and Umeda, 1993]. Particularly mechanical engineers have an image of an implementation in mind while talking about the abstract function. This may look like a disadvantage. On the other hand in the architecting phase behaviour is not fundamental [Szykman *et al.*, 1999]. Additionally, FunKey dissects the functions to be performed, and is independent of the method used to identify the functions (a mind map in the MAPPER case, functional block diagrams in the BOA case, various methods in the student project). The effect is that the right discussions between the system designers emerge automatically.

7.7 Conclusions and Recommendations

Due to limited automation, the architect's playground is not efficient at this moment. However, the strength of the method is shown both in finding new concepts, and in better information on the system architectures.

Some of the new concepts found are straightforward. Skilled engineers would have found them without using any tools. However, by using FunKey it is no longer a matter of chance whether they are found, and it appears time can be saved. In addition to these straight forward concepts, several unexpected and promising concepts have been found.

In addition to the well known functional models like functional block diagrams, mind maps are very handy with the FunKey analysis and design procedure.

In the near future we will create a computer-based prototype that supports the architect's playground to eliminate the book-keeping now done by hand. In a second version, the link with the innovative principles of TRIZ can be integrated. With this prototype, additional tests will be undertaken.

7.8 Acknowledgements

The author likes to thank the Master student Willem Alink for his work on the BOA case and MAPPER Lithography for the opportunity and their cooperation. Also the students in project DMS of 2007 are kindly thanked for their work.

Chapter VIII

Discussion, Conclusions and Recommendations

Now that we have a method and have results from its application, we can reflect on its use. This chapter will look at the place of the FunKey architecting method within the field of available methods. We will use the reference model derived in Chapter 4 for this purpose. Also we will reflect on the practical usefulness of FunKey, using the results of Chapter 7.
Finally the question whether we have achieved what we aimed at will be discussed. Conclusions and the directions this research can be continued finalise the chapter and the thesis.

The discussion in Section 8.1 will revisit the concept and system design process. Several observations will be made. The reference model, including the place of the developed FunKey tool within that model is treated. We will also reflect briefly on the results obtained with FunKey architecting. The final two sections contain the conclusions and recommendations for future research, respectively.

8.1 Discussion

8.1.1 Concept and system design

The early phase of the design process, where the system architecture is created, the basic concepts are chosen and the interfaces between the subsystems are defined, is crucial for the end result of the design project. The more care is taken in that phase, the smaller the number of occurring problems and required crash-actions in later phases.

Support in the conceptual phase has been very limited. There have been several attempts of letting the computer do the conceptual design by using Artificial Intelligence. While shown to work on class-room problems, they have not found much use in industry, partly because of lack of design databases that are required by Artificial Intelligence modules, partly because of fear by the designers. The approach taken in this thesis is therefore to use the real intelligence of the designers. By helping the designer(s) to organise and share their views and ideas, the knowledge base in the head(s) of the designer(s) will be easier to query. The expectation has been that in this way the fear of the designers will be reduced as they can continue doing the creative and inspiring part of the work, while the rest is being taken care for by the tool.

8.1.2 The conceptual and system design reference model

Using the product models that are used by experienced conceptual designers, we have developed a reference model for the conceptual phase of the design process. This model can be used for process-driven models and data-driven models for the conceptual design phase. The reference model has been used to compare several process models for the conceptual phase. In particular the connection from the process models to TRIZ is poor. While most

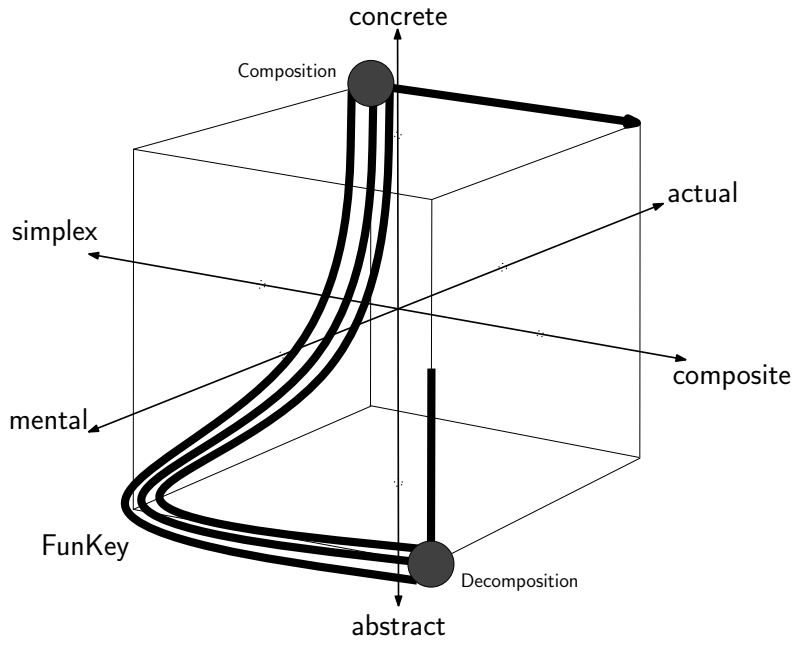


Figure 8.1: FunKey architecting and the reference model (see Figure 4.4, page 46).

process models describe *what* has to be done or is done in what sequence, there is often a lack of instructions of *how* to do that. TRIZ, on the other hand has many easily applied techniques that directly provide solutions.

8.1.3 FUNKEY AND THE REFERENCE MODEL

In Chapters 5 and 6 we proposed the FunKey architecting method to aid the system architect. To see the effect of FunKey architecting in the reference model, a few observations have to be made:

- Although we have given a series of steps for FunKey (page 60), it does not rely on the exact sequence of steps.
- The lists of key drivers and functions will be dynamical in the early part of the process. They serve as medium for discussion among the designers.
- The same holds, even stronger, for the coupling matrix $\mathcal{C}$.
- The FunKey process is hierarchical. Every level in the design builds upon the previous results. This is what we called *zooming in* in Chapter 5.

Figure 8.1 shows the route the FunKey method takes in the conceptual and system design space, for each hierarchical level. The filled circle at the bottom centre corner marked "Decomposition" shows how a complex problem is subdivided into smaller problems. FunKey helps in finding this mapping using the coupling matrix and the subsequently derived budgets. The other circle at the top rear corner marked "Composition" is the inverse:

it tells how the developed subsystems work in concert to reach the design goal. Using the composition, the effects of design decisions on any of the lower hierarchies can be investigated. As found in Chapter 5 the outcome of the composition should fit within the budget found in the decomposition (see Equation 5-4, page 64). It should be clear that FunKey does not prescribe the route. The designer can deviate from the route if needed.

As FunKey is inherently hierarchical, the result of the decomposition can be decomposed further. By using the inverse relation, the composition is directly known. This way, the entire intermediate level in Figure 1.1 can be covered in several layers of the FunKey tool. This can fill the gap shown in [Muller, 2004b] while simultaneously ensuring that both overview and detail information are available.

The detour in the "simplex" face of the reference model is the result of applying TRIZ in the systems architecting phase. The functions found in combination with the key drivers, are used to find applicable TRIZ principles. These principles then lead to solutions that can be integrated. Other routes from decomposition to composition may be added, like the method in [Tomiyama and Umeda, 1993]. At lower hierarchical levels the use of simulation tools like 20–Sim and matlab are possible. These can use the upper level parameters as input, the results are entered into the FunKey matrix to investigate the effects on higher levels. The connection with CAD, Finite Element Analysis tools and modelling with mechatronic features can also be researched.

The power of the FunKey method, as found in Chapter 7, lies in the fact that the decomposition and composition are developed simultaneously and can be extended to lower hierarchical levels. This results in the detail designer seeing where his part fits in the system, and providing the system designer with a tool for tracking the progress of the detail designers in the context of the entire system. This leads to improved understanding of the system. This is as required in Chapter 2, and corresponds to the goal we have set in Figure 1.2 on page 8.

8.1.4 FunKey and other conceptual design processes

Figure 4.5 (page 47) shows several design process models, mapped into the conceptual and system design space of the reference model. As concluded, there is no clear agreement on the starting position of the conceptual design phase. We can see that, except for TRIZ and the Vee-model, all process models have a route through the composite, abstract, mental corner. This corner is where FunKey creates the decomposition. Therefore FunKey can be connected to each of these design process models.

In the Krumhauer approach (Figures 2.5 and 4.5(a)), the FunKey route will connect to the goal as defined in Figure 2.5. In fact the decomposition that is sought in FunKey, and is defined as budgets for the key drivers, can be seen as a formalisation of the goal in Krumhauer's model.

When looking at the method by Pahl and Beitz (Figures 2.7 and 4.5(b)) and comparing that with the FunKey route, we can see that FunKey helps to find the solutions. The step called "Search for working principles to fulfil the subfunctions" gives a command, but does not provide information as to how to achieve it. With FunKey we have chosen to incorporate TRIZ as a means of achieving such goals. It provides guidance in finding solutions. Thus FunKey does not replace the Pahl and Beitz approach, it provides more insight into the

problem and helps in finding solutions, and providing for the insight in the hierarchy that is always present in system design.

As for TRIZ, there is no connection between TRIZ and the other design process models. FunKey fills that gap by creating a decomposition and directly providing leads to applicable TRIZ principles, as described in Chapter 6. Creativity of the concept and/or system designer is still required in defining the (de)composition and the translation of the abstract, generalised, TRIZ solutions into implementations suitable in the system at hand.

Christopher Alexander (Figures 2.2 and 4.5(d)) merely described design situations without giving a design process model. Nevertheless, we can see that FunKey can help in the self-conscious situation. It will provide a route through the design space that in the early and latest parts connects to the situation described by Alexander. In fact, it fits very well to the strategy of grouping the requirements based on the effect they have on (groups of) variables, as described by Alexander (see page 13).

The CAFCR model (Figures 2.4 and 4.5(e)) provides types of information that can be used in the FunKey approach. Starting from the Application and Functional view, the functions and key drivers are obtained that can be used in FunKey. The concretization that results in the Realization view in the CAFCR model, is not treated in the CAFCR model. FunKey helps in creating such realizations. One of the tools proposed is TRIZ, although, as said above, other methods and tools (simulation, CAD) may be of use, as well.

Another design method, related to the N^2 method in [Bustnay and Ben-Asher, 2005; INCOSE SEH Working Group, 2000], though not treated in previous chapters, is the Design Structure Matrix (DSM, see for instance [David M. Sharman, 2004; Eppinger, 1991] and `www.dsmweb.org`). Central in this approach is the square Design Structure Matrix where both the columns and rows represent "system elements". Relations between the system elements are marked with ones or crosses. By analysing the filled out DSM, relations between system elements are revealed. Structuring the design problem is achieved by rearranging the rows and columns so that the matrix becomes as diagonal as possible.

While in form comparable to FunKey, there is a significant difference in that FunKey looks at the more abstract functions (in the rows) and key drivers (in the columns), whereas the DSM uses system elements. These system elements are function bearers; not functions. Nevertheless, a connection between FunKey and DSM or N^2 seems appropriate.

8.1.5 Results of FunKey

There is more to FunKey than only connecting other design processes to TRIZ. As noted in Chapter 7, the designers with whom we have cooperated reported a better overview of, and insight into the system, while the effort to create and fill the coupling matrix is limited. The resulting increased understanding of the system, both for the system designers and detail designers, will reduce risks (see Section 4.5.1) and thus improve the development time and cost. Also, the coupling matrix forces the system designer to consider each cell. The chances of overlooking effects or contributions and opportunities for new solutions are therefore reduced.

In a practical design case, as experienced during the MAPPER case, and also from experience at ASML, the design process involves many people that work concurrently on different parts and/or aspects of the system under design. To ensure proper cooperation of all these parts, it is required that these concurrent designs are mutually well-adjusted. In many

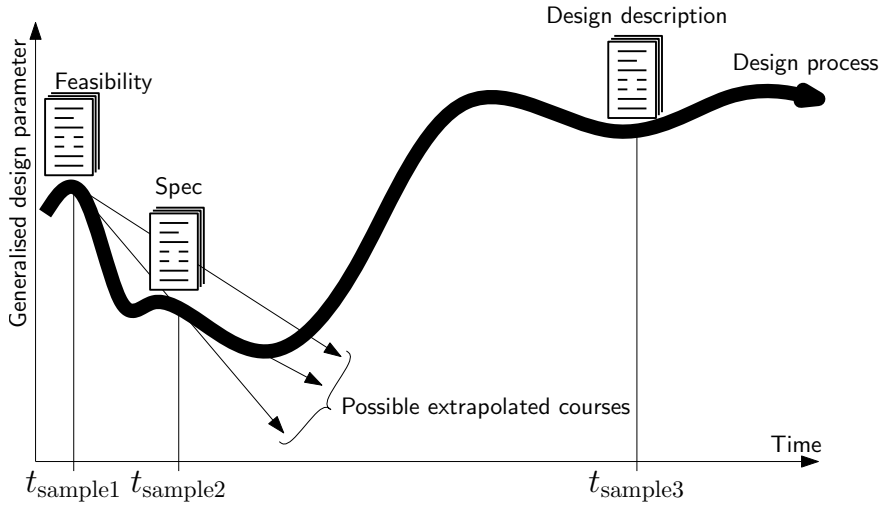


Figure 8.2: The course in a design project over time. The vertical axis is marked "generalised design parameter". This represents any aspect, like sensor resolution, number of sensors, power of the motor. The graph shows three "sample-moments": Feasibility study, Specification and Design description.

cases, there are only a limited number of moments at which these designs are reviewed, mostly by reviewing a document, see Figure 8.2. The feasibility study, or investigation of what is needed, the module specification and the design description are important milestones in the design process. According to the system engineering guidelines, these documents have to be reviewed in order to ensure a fitting subsystem. Referring to the Vee-model (Figure 4.3), the review moments for an entire system development are the system requirements, the allocation of functions to subsystems (and the accompanying specifications), and the components definition. Possibly some extra review moments are present.

One can say that the design process is *sampled* at these moments. Based on the first two reviews in Figure 8.2, representatives of adjacent modules, or the next higher hierarchical level, may deduct one of the courses marked "Extrapolated courses" for the future developments, depending on their assumptions and extrapolation algorithms. In reality though, the course is drastically adjusted shortly after the second review moment. Using terms from control engineering, we can state that due to the discretization of the design process by a documentation system, the process is under sampled. This will lead to erroneous results, just as under sampling a signal will prevent proper reconstruction.

The WWHWWW questions (Section 4.3) proposed in [Muller, 2004b] to sample the design process can provide for a higher sample rate. In software engineering the postlist contains a log of all modifications to the code. It is used to track which files are updated, including the reason. A close watch on the postlist by the system engineer may be a way to sample the design process when the system is software intensive. In other areas like mechanical or mechatronic engineering, such a mechanism is not present.

The FunKey method can be used to provide a mechanism to sample the design process at a higher rate: In a table like the one in Table 5.5, an extra column to accommodate for the

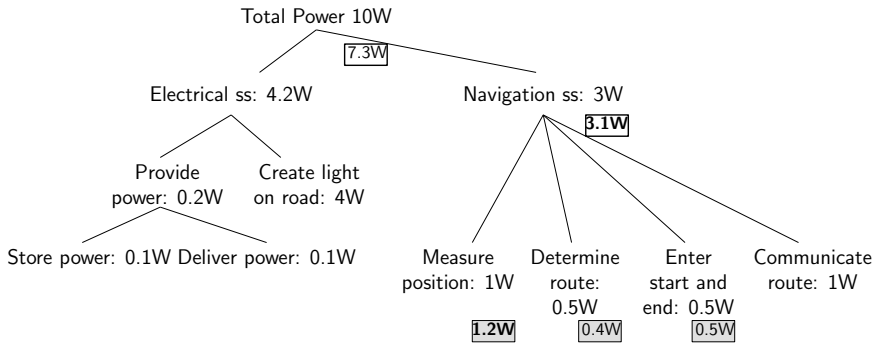


Figure 8.3: Power budget tree for the Personal Urban Transporter also shown in Figure 5.6. Now the bottom-up route is also shown.

actual numbers that follow from the design may suffice in combination with extra formatting showing compliance or non-compliance. When the budget is in a tree-form, like is shown in Figure 8.3, then two numbers can be shown, the *soll*-value and the *ist*-value (shown in boxes), again with different formatting for compliance (normal) and non-compliance (**boldface**). It may very well be possible that some of the numbers are available based on a (detail) design, while other numbers are not. Then, the bottom-up route may use available numbers where possible, and budgeted numbers (resulting from the top-down route) everywhere else. This is shown in Figure 8.3, where the actual value for Communicate route is not yet known. The upper level values (Navigation subsystem and Total power) thus use the budgeted value for calculating the ist-value. Also, reuse of numbers from previous versions/releases of the system may be used. Another possibility is when the budget tree is folded, and the system engineer only sees the upper few levels. He then knows the navigation subsystem uses too much power, but as the overrun is very small, he can decide to leave it as the margin is larger than the overrun.

In fact the author, when working at ASML, used this method in paper form. As a system engineer he discussed the overlay budget tree with designers, and noted the actual values in ink on the budget tree. This way he kept an eye on the developments and adjusted the budget when necessary.

The use of STFN's (as proposed in Section 5.5) may be of use here. If the numbers are not exactly known, a range can be given. The membership function then is a symmetrical triangle. The rules of addition and subtraction are very simple, as shown in Table 5.2. Using the navigation subsystem of the PUT of Figure 8.3, but now representing the expected outcomes of the detail process in STFNs, gives the budget tree in Figure 8.4. The total power consumption of the navigation subsystem is thus $[2.6, 5.5]$W. The membership function is also shown in Figure 8.4, together with the specification. As seen, the membership function contains the spec, However the chance of fulfilling the budget is small. Thus action is required by the system engineer. For instance, the STFNs for the Measure position and Communicate route should be reduced.

For this to work, the detail designers have to enter the resulting performance of their designs on a regular basis into the FunKey database. This requires some extra work on their part. The tool will not work when the interval between updates is too large. Possibly,

but this requires extra research, it is possible to devise automated coupling between the FunKey tool and CAD programs. Then, when the CAD model is updated, the data in the FunKey is updated as well. It is expected this is easily realized for parameters like weight. However, for performance related parameters like overlay in a wafer stepper, the automatic coupling is much more complicated. Whether the extra work to create the connection is counterweighted by the effort saved by it, remains to be seen. Also, automation is hard in cases where a technology change results in a significant change in performance. The experience and intuition of the designers may then be more valuable than calculations.

We aimed at (Section 1.3 on page 6) helping the designer in tedious and/or difficult tasks while using his creativity and capabilities as much as possible. At this moment the manual operations required to use FunKey are quite tedious. They can and will be automated. Then the designer will have more time to solve the crucial issues the method will help him to identify.

As can be seen from the concrete concepts generated using the FunKey tool (Sections 7.3.3 and 7.4.3), one can generate several feasible solutions for difficult problems by following the suggestions created with the coupling matrix. The TRIZ approach has been leading as it provides well described and easily applied patterns. We expect that incorporating more TRIZ tools will create an even richer and more powerful toolbox.

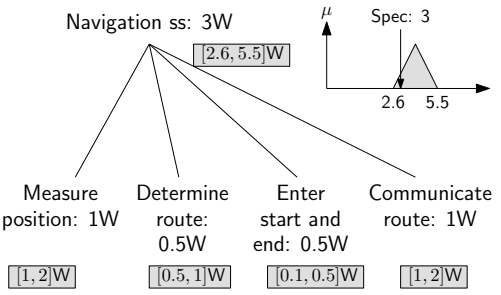


Figure 8.4: Power budget tree for the Navigation subsystem of the Personal Urban Transporter. The expected outcomes are represented with symmetrical triangular fuzzy numbers (STFNs).

8.2 Conclusions

The conceptual phase is the phase where many different views on the system design have to be integrated. Technical knowledge, production possibilities, market opportunities etc. all have to come together to define the new product. Designers active in that phase use several different types of product models to handle this information, and to communicate among each others. Also the communication with the marketing department is important. Models identified are 1. budgets, 2. mathematical and physics models, 3. block and functional diagrams, 4. specifications, 5. sketches, and 6. CAD. Where it should be noted that CAD is important when realizations are compared, not in the earliest phase of conceptual design.

The model of the conceptual phase, as described in [Krumhauer, 1974] (see Figure 2.5 on page 17), is used to order the models. There is a discrepancy between the models listed by the designers, and the recognition they have for the route through the space defined by Figure 2.5. Using this information, we have defined a reference model for the conceptual and system design phase. This model, shown in Figure 4.4 on page 46, is closely related to the model by Krumhauer but allows for more freedom. It is less prescriptive and can be used to compare process-driven and data-driven descriptions of the conceptual design phase.

FunKey, the *Fun*ction and *Key* driver based architecting tool that is developed, fits the design process models described in Chapters 2 and 4. It can be used to connect to the CAFCR model, the Krumhauer and Pahl & Beitz method. TRIZ is incorporated to help designers in finding innovative solutions. This way the large expertise developed in the

TRIZ community can be reused in designing complex systems. Not only that, FunKey also provides increased insight and overview into the complex system under design. As stated in the previous chapter, FunKey dissects the design problem. The dissection directly provides for the combination of subsolutions into one system. It is expected that the resulting improved understanding of the system under design, both at system design level and detail design level, will reduce the number and impact of risks.

The hierarchical approach in FunKey is able to provide overview where necessary and detail information where appropriate. Also the results of detail work can be used to track progress.

FunKey is a simple tool that can be learned quickly. As seen from the applications treated in Chapter 7, when applied from the beginning of a design project, it will yield:

- increased insight;
- improved understanding;
- better and more directed discussions among designers;

$\Rightarrow$ Insight

- documentation of decisions taken;
- tracking of progress of the design project over various hierarchical layers;

$\Rightarrow$ Overview

- easy access to innovative principles using TRIZ.

$\Rightarrow$ Innovation

To the right of these items, the connection to the goal set in Figure 1.2 (page 8) is shown. Clearly all aspects have been met. We therefore conclude that FunKey works and deserves to be developed further.

8.3 Recommendations

First priority now is to create a prototype tool that can be evaluated further in practical cases. This prototype tool will be created modularly so that the evaluation can become more and more complete. Proposed order of development is:

1. Basic FunKey tool that can be used to investigate functions and key drivers. The tool has to allow for
   - folding and unfolding of rows (functions) and columns (key drivers and requirements);
   - assignment of contributions of functions to key drivers. Where X-es are to be used as well as positive and negative contributions (+ and –), as well as excessive and insufficient contributions (e and i);
   - assignment of (sub)functions to subsystems within different architectures.
2. Export of the architectures to a format that can be used as skeleton of a budget.
3. Direct connection to TRIZ:
   - associate key drivers with the TRIZ parameters (automatic or manual);
   - based on the above associations, automatically suggest appropriate TRIZ principles using the priority matrices;
   - extend the connection to TRIZ with other TRIZ tools.

For the longer term we will direct our research to incorporate FunKey into QFD, and possibly the Design Structure Matrix/N^2, on the one hand, and integrating simulation tools (matlab/simulink, 20Sim) on the other hand. ASIT, in particular the Unification template, has some good chances of connecting FunKey to [Alink, 2007]. Finally, we will look at a connection with Axiomatic design, in particular to evaluate the created architectures.

Bibliography

Al-Salka, M., M. Cartmell and S. Hardy: 1998; *A Framework for a Generalized Computer-Based Support Environment for Conceptual Engineering Design*; **Journal of Engineering Design**; **vol. 9** (1): pp. 57–88.

Alexander, C.: 1966; Notes on the Synthesis of Form; Harvard University Press, Cambridge, Massachusetts, Cambridge (Mass.).

Alink, W.: 2007; Systematic, Conceptual Re-Design of a Waste Baler; Master thesis; University of Twente.

Altshuller, G. S.: 1997; 40 Principles - TRIZ Keys to Technical Innovation; *TRIZ Tools*, vol. 1; Technical Innovation Center, Worcester, MA.

Altshuller, G. S.: 1999; The Innovation Algorithm: TRIZ, Systematic Innovation and Technical Creativity; Technical Innovation Center, Worcester, MA.

van Amerongen, J. and P. C. Breedveld: 2003; *Modelling of Physical Systems for the Design and Control of Mechatronic Systems*; **Annual Reviews in Control**; **vol. 27** (1): pp. 87–117.
`http://www.sciencedirect.com/science/article/B6V0H-491J5GK-1/2/151935a8061aa8e6495e9a67a5cbd523`

de Andres Sanchez, J. and A. Terceno Gomez: 2003; *Applications of Fuzzy Regression in Actuarial Analysis*; **Journal of Risk and Insurance**; **vol. 70** (4): pp. 665–699.
`http://www.blackwell-synergy.com/doi/abs/10.1046/j.0022-4367.2003.00070.x`

Axelsson, J.: 2002; *Towards an Improved Understanding of Humans as the Components That Implement Systems Engineering*; in 12th International Symposium of the International Council on Systems Engineering; INCOSE, Las Vegas.

Blanchard, B. S. and W. J. Fabrycky: 1998; Systems Engineering and Analysis; Prentice Hall, Upper Saddle River, New Jersey; 3rd edn.

Bonnema, G. M.: 2006a; *Function and Budget Based System Architecting*; in TMCE 2006; pp. 1306–1318; Ljubljana, Slovenia.

Bonnema, G. M.: 2006b; *TRIZ for Systems Architecting*; in TRIZ Future 2006; pp. I: 87–92; Kortrijk, Belgium.

Bonnema, G. M. and F. J. van Houten: 2003; *Conceptual Design in a High-Tech Environment*; in 2003 International CIRP Design Seminar - "Methods and Tools for Co-operative and Integrated Design"; p. 84; Grenoble, France.

Bonnema, G. M. and F. J. van Houten: 2004; *Conceptual Design in a High-Tech Environment*; in Methods and Tools for Co-Operative and Integrated Design, edited by S. Tichkiewitch and D. Brissaud; pp. 171–182; Kluwer Academic Publishers, Dordrecht.

Bonnema, G. M. and F. J. van Houten: 2006; *Use of Models in Conceptual Design*; **Journal of Engineering Design**; **vol. 17** (6): pp. 549–562.
http://dx.doi.org/10.1080/09544820600664994

Bonnema, G. M., I. F. Lutters-Weustink and F. J. van Houten: 2005; *Introducing Systems Engineering to Industrial Design Engineering Students with Hands-on Experience*; in 18th International Conference on Systems Engineering (ICSEng05); pp. 408–413; IEEE Computer Society, Las Vegas.
http://ieeexplore.ieee.org/iel5/10446/33170/01562885.pdf?arnumber=1562885

Bracewell, R. H., D. A. Bradley, R. V. Chaplin, P. M. Langdon and J. E. E. Sharpe: 1993; *Schemebuilder, a Design Aid for the Conceptual Stages of Product Design*; in ICED '93; pp. 1311–1318; Heurista, Den Haag.

Bracewell, R. H. and J. E. E. Sharpe: 1996; *Functional Descriptions Used in Computer Support for Qualitative Scheme Generation - "Schemebuilder"*; **AI EDAM Journal - Special Issue: Representing Functionality in Design**; **vol. 10** (4): pp. 333–346.
http://www-edc.eng.cam.ac.uk/~rhb24/aiedam96.pdf

Bustnay, T. and J. Z. Ben-Asher: 2005; *How Many Systems Are There? – Using the N2 Method for Systems Partitioning*; **Systems Engineering, The Journal of the International Council on Systems Engineering**; **vol. 8** (2): pp. 109–118.

Buzan, T. and B. Buzan: 1995; The Mind Map Book; BBC Books, London; revised edn.

Calvano, C. N. and P. John: 2004; *Systems Engineering in an Age of Complexity*; **Systems Engineering, The Journal of the International Council on Systems Engineering**; **vol. 7** (1): pp. 25–34.
http://dx.doi.org/10.1002/sys.10054

Casti, J. L.: 1979; Connectivity, Complexity and Catastrophe in Large-Scale Systems; *International Series on Applied Systems Analysis*, vol. 7; John Wiley for the International Institute for Applied Systems Analysis, Chichester.

Chan, L.-K. and M.-L. Wu: 2002; *Quality Function Deployment: A Literature Review*; **European Journal of Operational Research**; **vol. 143** (3): pp. 463–497.
http://www.sciencedirect.com/science/article/B6VCT-45GKM72-3/2/c11f1f6c0eb065eb8191c6e59509e9a0

Chan, L.-K. and M.-L. Wu: 2005; *A Systematic Approach to Quality Function Deployment with a Full Illustrative Example*; **Omega**; **vol. 33** (2): pp. 119–139.
http://www.sciencedirect.com/science/article/B6VC4-4CJVRT5-2/2/8ce53973d4b82216c8ea5434e008ec11

Controllab Products: 2006; *20-Sim Homepage*; Last accessed on 14 March 2007.
http://www.20sim.com/

Daniels, J. and T. Bahill: 2004; *The Hybrid Process That Combines Traditional Requirements and Use Cases*; **Systems Engineering, The Journal of the International Council on Systems Engineering**; **vol. 7** (4): pp. 303–319.

David M. Sharman, A. A. Y.: 2004; *Characterizing Complex Product Architectures*; **Systems Engineering, The Journal of the International Council on Systems Engineering**; **vol. 7** (1): pp. 35–60.
http://dx.doi.org/10.1002/sys.10056

Deshmukh, A. V., J. J. Talavage and M. M. Barash: 1998; *Complexity in Manufacturing Systems Part 1: Analysis of Static Complexity*; **IIE Transactions**; **vol. 30** (7).
http://farm.ecs.umass.edu/papers/1998/IIE_TRANS/iie_trans.ps.gz

Dorée, A. G.: 1996; Gemeentelijk Aanbesteden - Een Onderzoek naar de Samenwerking Tussen Diensten Gemeentewerken en Aannemers in de Grond-, Weg- en Waterbouwsector; Ph.D.-thesis (in Dutch); University of Twente.

Driessen, K., P. Faber, K. Kooijman, P. Kramer, H. van Leeuwen and P. Schröder: 2006; *The FEI Small Dual Beam Product Family*; in Systems Research Forum, edited by R. Jain and B. Sauser; vol. 1; pp. 73–87; Schaefer School of Engineering Press; Stevens Institute of Technology, Hoboken, NJ.
http://www.stevens.edu/sse/fileadmin/sse/pdf/SystemsResearchForum.pdf

Dynasim: 2005; *Dymola*; Last accessed on 15 november 2005.
http://www.dynasim.se/dymola.htm

Eekels, J. and W. A. Poelman: 1995; Methodologie; *Industriële Productontwikkeling*, vol. 2; Lemma, Utrecht.

Eger, A. O., G. M. Bonnema, D. Lutters and M. C. van der Voort: 2004a; Productontwerpen; Lemma, Utrecht.

Eger, A. O., D. Lutters and F. J. van Houten: 2004b; *'Create the Future'*; in International Engineering and Product Design Education Conference; Delft, The Netherlands.

Eising, F.: 2007; *Als Je Het Begrijpt Kun Je Het Veranderen*; Inaugural address (in Dutch).

Eppinger, S. D.: 1991; *Model-Based Approaches to Managing Concurrent Engineering*; **Journal of Engineering Design**; **vol. 2** (4): pp. 283 – 290.
http://www.informaworld.com/10.1080/09544829108901686

Finger, S. and J. R. Dixon: 1989a; *A Review of Research in Mechanical Engineering Design. Part I: Descriptive, Prescriptive, and Computer-Based Models of Design Processes*; **Research in Engineering Design**; **vol. 1** (1): pp. 51–67.
http://dx.doi.org/10.1007/BF01580003

Finger, S. and J. R. Dixon: 1989b; *A Review of Research in Mechanical Engineering Design. Part II: Representations, Analysis, and Design for the Life Cycle*; **Research in Engineering Design**; **vol. 1** (2): pp. 121–137.
http://dx.doi.org/10.1007/BF01580205

French, M. J.: 1985; Conceptual Design for Engineers; Springer-Verlag, London.

Frericks, H., W. Heemels, G. J. Muller and J. Sandee: 2006; *Budget-Based Design*; in Boderc: Model-Based Design of High-Tech Systems, edited by W. Heemels and G. J. Muller; pp. 59–76; Embedded Systems Institute, Eindhoven.

Gelb, M. J.: 1998; How to Think Like Leonardo da Vinci: Seven Steps to Genius Everyday; Delacorte Press, New York.

Gelb, M. J.: 1999; Denken Als Leonardo da Vinci: Dagelijkse Genialiteit in Zeven Stappen; De Kern, Baarn.

Gell-Mann, M.: 1995; *"What Is Complexity?"*; **Complexity**; **vol. 1** (1): pp. 16–19.

Gero, J. S.: 1987; *Expert Systems in Computer-Aided Design: Proceedings of the Ifip Wg 5.2 Working Conference on Expert Systems in Computer-Aided Design*; in International Federation for Information Processing. Working Group 5.2–Computer-aided Design; North-Holland, Sydney (Jenolan Caves), Australia.

Goldenberg, J. and D. Mazursky: 2002; Creativity in Product Innovation; Cambridge University Press, Cambridge.

Govers, C. P. M.: 1996; *What and How About Quality Function Deployment (QFD)*; **International Journal of Production Economics**; **vol. 46-47**: pp. 575–585.
`http://www.sciencedirect.com/science/article/B6VF8-3VWC751-1P/2/4652fcea9c2f00ab64cf60e71d199ade`

Gulatti, R. K. and S. D. Eppinger: 1996; *The Coupling of Product Architecture and Organizational Structure Decisions*; *Working Paper 3906*; MIT Soal School of Management.
`http://web.mit.edu/eppinger/www/pdf/Gulati_SloanWP3906.pdf`

Heemels, W., L. Somers, P. van den Bosch, Z. Yuan, B. van der Wijst, A. van den Brand and G. J. Muller: 2006; *The Key Driver Method*; in Boderc: Model-Based Design of High-Tech Systems, edited by W. Heemels and G. J. Muller; pp. 27–42; Embedded Systems Institute, Eindhoven.

van der Heide, L.: 2007a; *A380 van Singapore Airlines Blijft de Enige dit Jaar*; **NRC Handelsblad**; **vol. 16 oktober 2007** (17).

van der Heide, L.: 2007b; *'Vliegtuig in Elkaar Zetten Blijkt Lastig'*; **NRC Handelsblad**; **vol. 26 oktober 2007** (11).
`http://www.nrc.nl/economie/article799292.ece/`

Hitchins, D. K.: 1992; Putting Systems to Work; John Wiley and Sons, Chichester, UK.

Hitchins, D. K.: 2000; Getting to Grips with Complexity or a Theory of Everything Else...; Derek K. Hitchins.
`http://www.hitchins.net/ASE_Book.html`

Horvath, I.: 2000; *Conceptual Design: Inside and Outside*; in International Seminar on Engineering Design on Integrated Product Development (Ediprod 2000); Zielona Gora, Poland.

Hurley, A.: 2004; *Not for the Practitioner (Review of the Art of Systems Architecting)*; Last accessed on 20 november 2006.
`http://www.amazon.com/exec/obidos/tg/detail/-/0849378362/102-3276509-9529734?v=glance`

INCOSE SEH Working Group: 2000; Systems Engineering Handbook; INCOSE; 2nd edn.
`http://66.34.135.97/membersonly2003/sehandbook/se_hdbk_v2.pdf`

Ivashkov, M. and V. Souchkov: 2004; *Establishing Priority of TRIZ Inventive Principles in Early Design*; in Design 2004; Dubrovnik.

Kals, H. J. J., C. A. v. Luttervelt and K. A. Moulijn (eds.): 1996; Industriële Productie; De Vey Mestdagh, Middelburg; 1st edn.

Kao, D. and N. Archer: 1997; *Abstraction in Conceptual Model Design*; **Int. J. Human-Computer Studies**; **vol. 46** (1): pp. 125–150.

Kawakami, K.: 1995; 101 Unuseless Japanese Inventions; HarperCollins.

Kroonenberg, H. v. d. and F. Siers: 1992; Methodisch Ontwerpen; Educaboek BV, Culemborg, The Netherlands; 1st edn.

Kruit, P.: 2007; *The Role of MEMS in Maskless Lithography*; **Microelectronic Engineering**; **vol. 84** (5–8): pp. 1027–1032.
`http://www.sciencedirect.com/science/article/B6V0W-4MYD60H-8/2/731ad800874bc6afc0f2d325c8df38d0`

Krumhauer, P.: 1974; Rechnerunterstützung für die Konzeptphase der Konstruktion: ein Beitrag zur Entwicklung eines Programmsystems für die Lösungsfindung Konstruktiver Teilaufgaben; Ph.D.-thesis; Technischen Universitaet Berlin.

Lennings, A., J. Broek, I. Horvath, W. Sleijffers and A. de Smit: 2000; *Editable Physical Models for Conceptual Design*; in Third International Symposium TMCE 2000; pp. 665–674; Delft, The Netherlands.

Lewes, G. H.: 1875; *The Problems of Life and Mind - First Series: The Foundations of a Creed*.

Loopstra, E. R., G. M. Bonnema, H. van der Schoot, G. P. Veldhuis and Y. B. P. Kwan: 1999; *Lithographic Apparatus Comprising a Positioning Device Having Two Object Holders*; European Patent Office; EP0900412.
`http://v3.espacenet.com/textdoc?DB=EPODOC&IDX=EP0900412&F=0`

Lutters, D.: 2001; Manufacturing Integration Based on Information Management; Ph.D.-thesis; University of Twente.

Lutters-Weustink, I. F., D. Lutters and F. J. van Houten: 2007; *Domain Integration by Means of Features*; in International Conference on Competitive Manufacturing; COMA'07; Stellenbosch.

Maier, M. W. and E. Rechtin: 2000; The Art of Systems Architecting; CRC Press, Boca Raton; 2nd edn.

Malmqvist, J.: 1994; *Computational Synthesis and Simulation of Dynamic Systems*; in Design Theory and Methodology Conference 1994; The American Society of Mechanical Engineers, New York, Minneapolis, MN, USA.
`http://w3.ppd.chalmers.se/~joma/publications/dtm94-paper.pdf`

Manson, S. M.: 2001; *Simplifying Complexity: A Review of Complexity Theory*; **Geoforum**; **vol. 32** (3): pp. 405–414.
`http://www.sciencedirect.com/science/article/B6V68-437XPXC-9/2/28058a2ba63bec48fa116cfa10520f23`

Martensson, L.: 1999; *Are Operators and Pilots in Control of Complex Systems?*; **Control Engineering Practice**; **vol. 7** (2): p. 173.
`http://www.sciencedirect.com/science/article/B6V2H-3W6MPG0-3/2/b8e816b29b8de013067494d0594ac5e5`

Mattos, N. M., K. Meyer-Wegener and B. Mitschang: 1993; *Grand Tour of Concepts for Object-Orientation from a Database Point of View*; **Data & Knowledge Engineering**; **vol. 9** (3): pp. 321–352.
`http://www.sciencedirect.com/science/article/B6TYX-47XF823-M/2/7d74a8df75236fb9486d717eb6c058c3`

McDavid, D.: 2005; *Systems Engineering for the Living Enterprise*; in ICSEng'05; pp. 244–249; IEEE Computer Society, Las Vegas.

McDermid, J. A.: 2000; *Complexity: Concept, Causes and Control*; in Sixth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2000); IEEE.

Mekid, S. and M. Bonis: 1997; *Conceptual Design and Study of High Precision Translational Stages: Application to an Optical Delay Line*; **Precision Engineering**; **vol. 21** (1): pp. 29–35.
`http://www.sciencedirect.com/science/article/B6V4K-3SNMSKB-3/2/3b5b4f5cb61bb598c49d5795cc9495af`

Miller, G. A.: 1956; *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information*; **Psychological Review**; **vol. 63**: pp. 81–97.
`http://psychclassics.yorku.ca/Miller/`

Mital, A., M. Kulkarni, R. Huston and S. Anand: 1997; *Robot Cleaning of Underground Liquid Storage Tanks: Feasibility and Design Considerations*; **Robotics and Autonomous Systems**; **vol. 20** (1): pp. 49–60.
`http://www.sciencedirect.com/science/article/B6V16-3TCMS0C-4/2/4d139a468c6bf0f503ef27a20233c6f0`

Muller, G. J.: 2004a; *Architecting for Humans, How to Transfer Experience?*
`http://www.gaudisite.nl/ExperienceTransferSlides.pdf`

Muller, G. J.: 2004b; CAFCR: A Multi-View Method for Embedded Systems Architecting; Ph.D.-thesis; Delft University of Technology.
`http://www.gaudisite.nl/Thesis.html`

Muller, G. J.: 2004c; *The Waferstepper Challenge: Innovation and Reliability Despite Complexity*.
http://www.gaudisite.nl/IRCwaferstepperPaper.pdf

Muller, G. J.: 2005a; *The Arisal of a System Architect*; Last accessed on 09 november 2005.
http://www.gaudisite.nl/MaturityPaper.pdf

Muller, G. J.: 2005b; *Do Useful Multi-Domain Methods Exist?*; in Conference on Systems Engineering Research (CSER); Hoboken, NJ, USA.

Muller, G. J.: 2007; *Design Objectives and Design Understandability*; Last accessed on 7 june 2007.
http://www.gaudisite.nl/DesignUnderstandabilitySlides.pdf

Nakagawa, T.: 2006; *A New Paradigm for Creative Problem Solving: Six-Box Scheme in Usit*; in TRIZ Future 2006; pp. II: 45–50; Kortrijk, Belgium.
http://www.triz-journal.com/archives/2007/01/06/

Nakagawa, T. and K. Yasui: 2003; *Note on Reliability of a System Complexity*; **Mathematical and Computer Modelling**; **vol. 38** (11-13): p. 1365.
http://www.sciencedirect.com/science/article/B6V0V-4BRR7B3-11/2/18feadf8ba6036ce432051e061d49fd1

NRC: 2007; *Eerste Airbus A380 Geleverd, Anderhalf Jaar Te Laat*; **NRC Handelsblad**; **vol. 15 oktober 2007**: p. 11.
http://www.nrc.nl/economie/article788262.ece/

Ottino, J. M.: 2003; *Complex Systems*; **AIChE Journal**; **vol. 49** (2): p. 292.
http://www.sciencedirect.com/science/article/B6WR2-47W6584-1/2/c31c3b1ed6be78e3a6ee49e0556adfa4

Ottoson, S.: 1998; *Qualified Product Concept Design Needs a Proper Combination of Pencil-Aided Design and Model-Aided Design before Product Data Management*; **Journal of Engineering Design**; **vol. 9** (2): pp. 107–119.
http://journalsonline.tandf.co.uk/openurl.asp?genre=article&id=doi:10.1080/095448298261570

Pahl, G. and W. Beitz: 1996; Engineering Design - a Systematic Approach; Springer-Verlag, London.

Pugh, S.: 1991; Total Design: Integrated Methods for Successful Product Engineering; Addison-Wesley, Amsterdam.

QFD Institute: 2005; *The Official Source for QFD*; Last accessed on 16 december 2005.
http://www.qfdi.org/

Ruder, J. A. and D. K. Sobek: 2007; *Experiment on a System-Level Design Tool*; **Journal of Engineering Design**; **vol. 18** (4): pp. 327 – 342.
http://dx.doi.org/10.1080/09544820601008928

Sage, A. P. and J. E. Armstrong jr.: 2000; Introduction to System Engineering; John Wiley and Sons, Inc., New York.

Salomons, O. W.: 1995; Computer Support in the Design of Mechanical Products; Ph.D.-thesis; University of Twente.

Salter, A. and D. Gann: 2003; *Sources of Ideas for Innovation in Engineering Design*; **Research Policy**; **vol. 32** (8): p. 1309.
`http://www.sciencedirect.com/science/article/B6V77-47CBD1W-3/2/5b2ed12a3a63f29661bc75316bd6f3a3`

Schön, D. and J. Bennet: 1996; *Reflective Conversation with Materials*; in Bringing Design to Software, edited by T. Winograd; pp. 171–184; ACM Press, New York.
`http://hci.stanford.edu/bds/9-schon.html`

Schulz, A. P. and E. Fricke: 1999; *Incorporating Flexibility, Agility, Robustness, and Adaptability within the Design of Integrated Systems - Key to Success?*; in Digital Avionics Systems Conference, 1999. Proceedings. 18th; vol. 1.
`http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=863677`

Shimomura, Y., H. Takeda, M. Yoshioka, Y. Umeda and T. Tomiyama: 1995; *Representation of Design Object Based on the Functional Evolution Process Model.*; in Design Theory and Methodology – DTM'95; pp. 351–360; ASME.
`http://www-kasm.nii.ac.jp/papers/takeda/95/dtm95.pdf`

Suh, N. P.: 1990; The Principles of Design; Oxford Series on Advanced Manufacturing; Oxford University Press, New York, Oxford.

Szykman, S., J. W. Racz and R. D. Sriram: 1999; *The Representation of Function in Computer-Based Design*; in ASME Design Engineering Technical Conferences; ASME, Las Vegas, Nevada.

Takasugi, S., Y. Miura and E. Arai: 1996; *Conceptual Design of an Electromagnetic Tomography System*; **Journal of Applied Geophysics**; **vol. 35** (2-3): pp. 199–207.
`http://www.sciencedirect.com/science/article/B6VFC-3WDC3K2-P/2/d635930ae175e092f7e27a3c2e0925b0`

Taleb-Bendiab, A.: 1993; *Conceptdesigner: A Knowledge-Based System for Conceptual Engineering Design*; in ICED '93; pp. 1303–1310; Heurista, Den Haag.

Tomiyama, T. and Y. Umeda: 1993; *A CAD for Functional Design*; **Annals of CIRP**; **vol. 42** (1): pp. 143–146.

de Vries, T. J. A.: 1994; Conceptual Design of Controlled Electro-Mechanical Systems : A Modeling Perspective; Ph.D.-thesis; Universiteit Twente.

Wojtkowski, W. and W. G. Wojtkowski: 2002; *Storytelling: Its Role in Information Visualization*; in The fifth European Systems Science Congress, 16-19 October 2002; Association Française de Science des Systèmes Cybernétiques, Cognitifs et Techniques; Systems Science European Union, Crete, Greece.
`http://www.res-systemica.org`

Zuo, J. and S. W. Director: 2000; *An Integrated Design Environment for Early Stage Conceptual Design*; in Design, Automation and Test in Europe Conference and Exhibition 2000. Proceedings.
`http://ieeexplore.ieee.org/iel5/6761/18074/00840887.pdf?tp=&isnumber=&arnumber=840887`

Zurro, B., C. Burgos, A. Ibarra and K. J. McCarthy: 1997; *Conceptual Design of a Phosphor Detector for an Iter-Like Plasma*; **Fusion Engineering and Design**; **vol. 34-35**: pp. 353–357.
`http://www.sciencedirect.com/science/article/B6V3C-3SN5TJ5-27/2/2067005dd6ef0dcac8c9aa329c1fc13a`

Zwikker, R.: 2007; *Organisation of Project Teams at Demcon; Personal Communications*.

The URLs shown in several references link to the cited source. To use these links, the information provider may require registration or a valid subscription.

Appendices

Appendix A

Questionnaire Used to Identify Product Models Used by Conceptual Designers

In Chapter 3 a questionnaire has been developed, issued and analysed to investigate the product models that conceptual designers use. On the following pages the questionnaire as issued is shown.

Dear Colleague,
At the Laboratory of Design, Production and Management of the Department of Engineering Technology of the University of Twente, several research projects are carried out that deal with supporting design; one of these projects concerns supporting conceptual design and system architecting.
In the conceptual phase, early in the design process, engineering science, practical knowledge, production methods, and commercial aspects need to be brought together, and the most important decisions are taken [French1985]. This means that aspects from very different disciplines have to be considered. This is among others done by using several types of models.
In order to get insight into the models used by experienced conceptual designers, this survey is done. The outcome will be used in the project that eventually will lead to a (prototype) support system for conceptual design and system architecting.
I would be very thankful if you, as an experienced conceptual designer, would take some 15 minutes of your time to fill out the questions below and e-mail (or fax) the answers to me. Also, if you have fellow-conceptual designers, I would be very thankful if you could send them this questionnaire as well.
With many thanks and kindest regards,
Ir. G.Maarten Bonnema
OPM/CTW
University of Twente
Tel: +31 53 489 2548
Fax: +31 53 489 3631
g.m.bonnema@utwente.nl

Practical information

Time required: approximately 15 minutes
Number of questions: 15
Reply to: g.m.bonnema@utwente.nl or Fax:+31 53 489 3631
The information you provide will not be used for any other purpose than the research project mentioned above.

Personal information

1. What is your age ___
2. What was your last education _____
3. How many years do you have working experience as:
    a. Designer ___
    b. Conceptual designer ___

Work content and environment

4. What type of industry do you work in: _____
5. How many fellow-conceptual designers do you have: ___
6. Please describe the type of problems you need to solve in your work:

_____
_____
_____
_____

7. Rate the problems you need to solve along the scales Complexity, Concreteness and Realisation. (1: least, 5: most; you can also indicate a range e.g. (4-5)):
   a. **Complexity**:

Single device	–	Chain of devices	–	Structure of devices
1	2	3	4	5

   b. **Concreteness**:

Physical principle	–	Working principle	–	Construction element
1	2	3	4	5

   c. **Realisation**:

Idea	–	Technical drawings	–	Product
1	2	3	4	5

Models

This Section of the survey asks you to list the models you use to solve the design problems you are confronted with. If you use many different models in a category, state at least the most important ones.
In addition to mentioning the models, also rate the concreteness, complexity and realisation of the corresponding problems on a 1-5 scale (1: least, 5: most; see the scales mentioned in question 7).
It would be highly appreciated if you can provide examples of these models (please attach).

8. What **textual list-like** models do you use? (Specifications, Budgets etc.)

(Complexity/Concreteness/Realisation)

_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)

9. What **textual descriptive** models do you use? (Scenario's etc.)

(Complexity/Concreteness/Realisation)

_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)

10. What other textual models do you use?

(Complexity/Concreteness/Realisation)

_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)

11. What **abstract graphical** models do you use? (Block-diagrams, (electrical) schematics, flow diagrams etc.):

(Complexity/Concreteness/Realisation)

_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)

12. What **concrete graphical** models do you use? (Sketches, CAD-models etc.)

(Complexity/Concreteness/Realisation)

_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)

13. What other graphical models do you use?

(Complexity/Concreteness/Realisation)

_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)

14. What **analytical** models do you use? (Mathematical etc.)

(Complexity/Concreteness/Realisation)

_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)
_____ (___/___/___)

15. Please describe any other types of models you use

(Complexity/Concreteness/Realisation)

_____ (___/___/___)

_____ (___/___/___)

_____ (___/___/___)

_____ (___/___/___)

Remarks

Please feel free to make any remarks about this survey and/or about the models you have mentioned. If the space provided is insufficient, please add extra space as you like.

_____

_____

_____

_____

If you would like more information, you can indicate it here, or you can contact me at the address given in the Introduction.

- o Yes, I would like more information about the outcome of the survey.
- o Yes, I would like more information about the research project
- o I would like to know more about:

_____

If you marked one or more options above, please enter your name and address:

_____

_____

_____

_____

Thank you very much for your time and effort.

References

[French1985] French, Michael; *Conceptual Design for Engineers*; The Design Council London; Springer-Verlag; ISBN: 0-85072-155-5

Appendix B

Symmetrical Triangular Fuzzy Numbers

B.1 Introduction

In this appendix, the arithmetic of Symmetrical Triangular Fuzzy Numbers (STFN) is treated. First, the basic form of an STFN is treated. Then two forms to represent an STFN are discussed, including the arithmetic to convert one representation into the other and vice versa. The goal of this appendix is to derive the arithmetic that is needed in system budgets. This means deriving the arithmetic for:

- addition;
- subtraction;
- scalar multiplication; and
- RMS addition.

When these operations are known, all relevant budgets can be expressed in STFNs.

B.2 STFN basics

As the name expresses, an STFN is a fuzzy number with a symmetrical triangular membership function $\mu(x)$, see Figure B.1. In non-fuzzy mathematics, membership is either true (1), or false (0). In fuzzy mathematics, there is a gradual change from true to false. This is described by the membership function. The membership function can have many shapes as long as there is at least one x-value for which $\mu(x) = 1$. In this thesis, only symmetrical triangular fuzzy numbers are used. Then, the membership function is triangular, and the left spread (l_L) and right spread (l_R) are equal, as shown in Figure B.1.

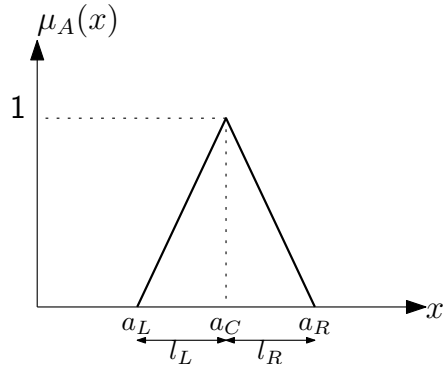


Figure B.1: Membership function in case of Symmetrical Triangular Fuzzy Numbers.

An STFN can now be expressed by the position and width of the triangular membership function. [Chan and Wu, 2005] use the left and right edge of the triangle to represent the fuzzy number:

$$A = [a_L, a_R] \tag{B-1}$$

Note the use of square brackets in this representation of an STFN. As the membership function is symmetrical, the center a_C can easily be calculated:

$$a_C = \frac{a_L + a_R}{2} \tag{B-2}$$

[de Andres Sanchez and Terceno Gomez, 2003] represent an STFN using the center and spread value:

$$A = (a_C, l_A) \tag{B-3}$$

Where $l_A = l_L = l_R$. Note the use of round brackets. The left and right edges can be calculated using:

$$a_L = a_C - l_A \tag{B-4}$$

$$a_R = a_C + l_A \tag{B-5}$$

Using the equations B-2, B-4 and B-5, the notations used in [Chan and Wu, 2005] can be easily converted to notation used in [de Andres Sanchez and Terceno Gomez, 2003], and vice versa:

$$[a_L, a_R] = [a_C - l_A, a_C + l_A] \tag{B-6}$$

$$(a_C, l_A) = \left(\frac{a_L + a_R}{2}, \frac{a_R - a_L}{2} \right) \tag{B-7}$$

where it is assumed that $a_R \geq a_L$, as illustrated by Figure B.1.

In this thesis the notation in [Chan and Wu, 2005] is used normally, as indicated by the use of square brackets. Only when needed the alternative notation in [de Andres Sanchez and Terceno Gomez, 2003] is used.

B.3 STFN Arithmetic

Addition, subtraction and scalar multiplication are derived both in [Chan and Wu, 2005] and [de Andres Sanchez and Terceno Gomez, 2003]. It is easy to derive that the two references agree on addition:

$$[a_L, a_R] + [b_L, b_R] = [a_L + b_L, a_R + b_R] \tag{B-8}$$

As for scalar multiplication, [Chan and Wu, 2005] only mention multiplication with a scalar $k > 0$. This is due to the fact that for $k < 0$ the left and right borders switch places. Using the centre and spread, as in [de Andres Sanchez and Terceno Gomez, 2003], is therefore more appropriate:

$$k \times (a_C, l_A) = (ka_C, |k| l_A) \tag{B-9}$$

From this and equation B-6, scalar multiplication for all values of k can be derived:

$$k \times [a_L, a_R] = \begin{cases} [ka_L, ka_R] & \text{if } k \geq 0 \\ [ka_R, ka_L] & \text{if } k < 0 \end{cases} \tag{B-10}$$

Then, by noting that subtraction is addition (equation B-8), combined with scalar multiplication where $k = -1$ (equation B-10), subtraction is calculated by:

$$[a_L, a_R] - [b_L, b_R] = [a_L - b_R, a_R - b_L] \tag{B-11}$$

Note the difference with the rule in [Chan and Wu, 2005]. This difference is probably due to the fact that scalar multiplication with $k < 0$ has not been defined in [Chan and Wu, 2005].

Finally, the arithmetic rule for RMS addition is needed for calculating budgets (see equation 5-3 on page 64). This, however, is a non-linear operation on STFNs. Such operations in general do not result in an STFN. Fortunately, an approximation is possible using the notation (a_C,l_A) as shown in [de Andres Sanchez and Terceno Gomez, 2003]:

$$(a_C, l_A)^p \approx \left(a_C^p, |p| a_c^{p-1} l_A\right) \tag{B-12}$$

Thus calculating the square root and the square are possible:

$$\sqrt{(a_C, l_A)} = (a_C, l_A)^{\frac{1}{2}} \approx \left(\sqrt{a_C}, \frac{l_A\sqrt{a_c}}{2a_C}\right) \tag{B-13}$$

$$(a_C, l_A)^2 \approx \left(a_C^2, 2a_c l_A\right) \tag{B-14}$$

In $[a_L,a_R]$ notation, these equations can be found using equations B-6 and B-7:

$$\sqrt{[a_L, a_R]} \approx \left[\frac{\sqrt{2}(3a_L + a_R)}{4\sqrt{a_L + a_R}}, \frac{\sqrt{2}(3a_R + a_L)}{4\sqrt{a_L + a_R}}\right] \tag{B-15}$$

$$[a_L, a_R]^2 \approx \left[\frac{(3a_L - a_R)\,(a_L + a_R)}{4}, \frac{(3a_R - a_L)\,(a_L + a_R)}{4}\right] \tag{B-16}$$

Combining equations B-8, B-15 and B-16, the RMS summation of two STNFs can be derived:

$$[c_L, c_R] = \sqrt{[a_L, a_R]^2 + [b_L, b_R]^2} \approx \tag{B-17}$$

$$\approx \left[\frac{a_L(a_L + a_R) + b_L(b_L + b_R)}{\sqrt{(a_L + a_R)^2 + (b_L + b_R)^2}}, \frac{a_R(a_L + a_R) + b_R(b_L + b_R)}{\sqrt{(a_L + a_R)^2 + (b_L + b_R)^2}}\right] \tag{B-18}$$

This completes the arithmetic required for system budgets. The rules are summarized in Table 5.2 on page 61.

B.4 Conclusions

When using Symmetrical Triangular Fuzzy Numbers, the operations needed for compiling and deriving a system budget are developed. Although approximation has been needed for calculating the square and the square root, the resulting arithmetic rules comply with arithmetic for normal numbers. For instance, calculating using equation B-18, the RMS sum of the degenerated STFNs [3,3] and [4,4] yields [5,5], as expected. The RMS sum of [2,4] ("approximately three") and [3,5] ("approximately four") gives $\left[\frac{18}{5}, \frac{32}{5}\right]$ ("approximately five").

Appendix C

TRIZ Priority Matrices

C.1 Introduction

Based on the work described in [Ivashkov and Souchkov, 2004], the priorities of TRIZ principles have been determined. This has been done both for determining the most appropriate principles to *improve* a technical parameter (Section C.2) and for *reducing the negative impact* on a technical parameter (Section C.3). The former is directly based on the work in the earlier mentioned reference [Ivashkov and Souchkov, 2004], whereas the latter is analogously.

For the positive priority matrix PM+ (Section C.2), the number of occurrences of a principle on a row in the TRIZ contradiction matrix is counted. The principles with the highest occurrences are the favourite principles for that technical parameter. In addition to the results presented in [Ivashkov and Souchkov, 2004] we have added two more columns for the second and third best principles for each technical parameter.

For some technical parameters the second and/or third best principles are blank. This indicates that the difference between the occurrence of the favourite and second or the second and third best principles was larger than 4 or 3, respectively.

The procedure for determining the negative priority matrix PM- (Section C.3) is identical. The only difference is that now the columns are analysed, instead of the rows.

Finally, the 40 Innovative Principles are listed in Appendix C.4. This way, all information needed to perform the TRIZ-related procedures in this thesis is present, except for the contradiction matrix [Altshuller, 1997].

C.2 *Positive Priority Matrix, PM^+*

	Technical Parameter	1st	2nd	3d
1	Weight of moving object	35		28
2	Weight of stationary object	35	10, 19, 28	1
3	Length of moving object	1, 29	15, 35	4
4	Length of stationary object	35	28	1, 10, 14, 26
5	Area of moving object	2, 15	13, 26, 30	4
6	Area of stationary object	18	2	35
7	Volume of moving object	1, 35	2, 10, 29	4, 15, 34
8	Volume of stationary object	35		
9	Speed	28	35	13
10	Force (Intensity)	35	10, 18, 37	36
11	Stress or pressure	35	10	
12	Shape	10	1, 14, 15	32, 34
13	Stability of the object's composition	35		
14	Strength	3, 35	10	40
15	Duration of action of moving object	19	35	3, 10
16	Duration of action of stationary object	35	1, 10, 16	40
17	Temperature	35	19	
18	Illumination intensity	19	32	
19	Use of energy by moving object	35	19	
20	Use of energy by stationary object	19, 35	18, 27	many
21	Power	35	19	2, 10
22	Loss of energy	7	35	2
23	Loss of substance	10	35	
24	Loss of information	10	26, 35	
25	Loss of time	10	35	18
26	Quantity of substance/the matter	35		
27	Reliability	35	11	10
28	Measurement accuracy	32	28	
29	Manufacturing precision	32		
30	Object-affected harmful factors	22	35	
31	Object-generated harmful factors	22, 35	2	1, 39
32	Ease of manufacture	1		
33	Ease of operation	1	13	2
34	Ease of repair	1	10	
35	Adaptability or versatility	35		
36	Device complexity	13, 26	1, 28	2, 10
37	Difficulty of detecting and measuring	28	35	16, 18, 26, 27
38	Extent of automation	35	13	28
39	Productivity	10	35	

C.3 Negative Priority Matrix, PM⁻

	Technical Parameter	1st	2nd	3d
1	Weight of moving object	35	8, 15, 28	2, 26, 40
2	Weight of stationary object	35	26, 27	
3	Length of moving object	1	15	14, 17, 28, 29
4	Length of stationary object	14, 26	1, 10, 35	7, 15, 28
5	Area of moving object	17	13, 15, 26	10, 29
6	Area of stationary object	16, 18, 35, 40	2, 10, 17, 39	30, 36
7	Volume of moving object	35	2, 10	29
8	Volume of stationary object	35		
9	Speed	35	28	
10	Force (Intensity)	35	10	
11	Stress or pressure	35	10	
12	Shape	1, 35	29	10, 14, 15
13	Stability of the object's composition	35		
14	Strength	28	3	10
15	Duration of action of moving object	35	3	19
16	Duration of action of stationary object	16	10, 35	1
17	Temperature	35	19	
18	Illumination intensity	19	32	1, 13
19	Use of energy by moving object	35	19	
20	Use of energy by stationary object	1	35	18, 19
21	Power	35	10, 19	2, 18
22	Loss of energy	35	2	
23	Loss of substance	10	35	
24	Loss of information	10	24	
25	Loss of time	10	35	28
26	Quantity of substance/the matter	35	3	
27	Reliability	10	11	40
28	Measurement accuracy	28	32	
29	Manufacturing precision	32	10	28
30	Object-affected harmful factors	22	35	2
31	Object-generated harmful factors	35	2	
32	Ease of manufacture	1	35	
33	Ease of operation	1	32, 35	2, 28
34	Ease of repair	1	10	
35	Adaptability or versatility	15		
36	Device complexity	1	26	10
37	Difficulty of detecting and measuring	35		
38	Extent of automation	35	2	
39	Productivity	35		

C.4 TRIZ 40 Principles

This appendix contains the 40 principles as described in more detail in [Altshuller, 1997].

#	Principle
1	Segmentation
2	Extraction
3	Local quality
4	Asymmetry
5	Consolidation
6	Universality
7	Nesting (Matrioshka)
8	Counterweight
9	Prior counteraction
10	Prior action
11	Cushion in advance
12	Equipotentiality
13	Do it in reverse
14	Spheroidality
15	Dynamicity
16	Partial or excessive action
17	Transition into a new dimension
18	Mechanical vibration
19	Periodic action
20	Continuity of useful action

#	Principle
21	Rushing through
22	Convert harm into benefit
23	Feedback
24	Mediator
25	Self-service
26	Copying
27	Dispose
28	Replacement of mechanical system
29	Pneumatic or hydraulic construction
30	Flexible films or thin membranes
31	Porous materials
32	Changing the colour
33	Homogeneity
34	Rejecting and regenerating parts
35	Transformation of properties
36	Phase transition
37	Thermal expansion
38	Accelerated oxidation
39	Inert environment
40	Composite materials

Appendix D

N^2 Diagrams from the DMS project

In this appendix, two N^2 diagrams, made by students in project Design of Mechatronics and Systems (Section 7.5) are shown. In an N^2 diagram, the subsystems are placed on the diagonal of a square $N \times N$ matrix. The outputs from a subsystem, say S1, are on the row of that subsystem S1. The inputs to a subsystem S2 are in the column of that subsystem S2. Thus an interface between subsystems S1 and S2 is found in the cell in row S1 and column S2 when it is an output of S1, and in the cell in row S2 and column S1 when it is an output of S2 (and thus input to S1). An N^2 diagram can also be used for identifying the interfaces between functions.

The first N^2 diagram (Table D.1) was created by Team C, that applied the FunKey method. The second example (Table D.2) was created by a team (Team A) that did not use FunKey to its full extent.

The N^2 diagram in Table D.1 is the top-level view of the diagram. The choice of the subsystems (Software, Puppets, Ball, and User Interface) is so that the interfaces between them are scarce. Each of the subsystems was worked out with corresponding N^2 diagrams. These were filled somewhat more. This shows that the choice of subsystems was done well; little interaction *between* the subsystems, more interaction *within* the subsystems.

The N^2 diagram in Table D.2 is filled far more than the one in Table D.1 and thus there are many interfaces between subsystems. Also, several signals occur more than once in the diagram. This can be necessary, but the large number of repetitions suggests improvements are possible.

Table D.1: N^2 diagram showing the system architecture of Team C.

Software	-Data: position -Data: if goal	-Data: if new game	
-Data: position	**Puppets**		
-Data: position		**Ball**	-Data: if goal
-Data -1 package before game			**User Interface**

Table D.2: N^2 diagram showing the system architecture of Team A.

Ball		-Ball pos. (x,y)		-Selfcheck report -Update report
	Player figures	-Player figures (x,Θ) pos. (system&user)		-Selfcheck report -Update report
	-Stop moving figures	**Maintain rules**	-Player figures (x,Θ) pos. (system&user) -Ball pos. (x,y)	-Selfcheck report -Update report -All figure data (x,Θ) -Ball pos. (x,y) -Game signals
	-Figures rot. (v,a) -Figures x-pos. (v,a)		**Strategy**	-Selfcheck report -Update report
-Start selfcheck -Update	-Start selfcheck -Update	-Start selfcheck -Update -Input by user -Game settings	-Start selfcheck -Update -Game settings	**User Interface**

About the Author

Gerrit Maarten Bonnema was born on April 23rd, 1968 in Hengelo. After attending primary school and the first year of secondary school in his hometown, he went to the Ichthus College in the neighbour town Enschede. There he followed the Gymnasium for two years. He then switched to the Atheneum, abandoning the classical languages Latin and Greek. Physics became his favourite subject. After finishing the Atheneum, he started his studies in Electrical Engineering at the University of Twente. He finished his studies in August 1992 with a Master thesis on the performance of a piezoelectrically driven one-port resonator. This MEMS related research involved modelling and measuring several vibrating microbeams.

During the Master studies, the interest in a broader education than Electrical Engineering arose. He therefore continued his studies in Delft, in a post-MSc course on Technical Systems. This course involved theory from Mechanical Engineering and a practical project. During the project he developed, together with a Master student, a compact XY($\Phi = 0$)-stage. The mechanics, measurement and control strategy were developed using an integrated approach; thus practising mechatronic design.

In 1995 he started at ASML, now the world largest wafer stepper supplier, as a systems engineer. His main focus was on the mechatronics of the wafer and reticle stages, and the overlay for the TwinScan® wafer scanners. Contributions were made to dose-control and control architecture, as well. In this role he contributed to several patents.

In 1999 he started as an assistant professor at the University of Twente. In this position he contributed to the design education of the Mechanical Engineering curriculum. He also was one of the members of the team starting the new curriculum for Industrial Design Engineering. For this curriculum he wrote, together with three colleagues, an introductory book on product design. In parallel to teaching, he researched the conceptual design process, and in particular the system design of complex systems. The present thesis is a result from this work.

While employed at the University of Twente, the other Dutch wafer stepper manufacturer, MAPPER lithography, hired him to contribute to their system design; in particular to the wafer positioning system. This fitted very well into his research, as can be seen in this thesis.

During the post MSc course, he met Lilian in Switzerland at a ski-camp. They married in Oberhofen, Switzerland in 1996. Now they have two sons: Joris (2000) and Casper (2002). As a family they like the outdoors. Camping and hiking are favourite past-times.

Index