Design with Overview – how to survive in complex organisations

G. Maarten Bonnema P. Daniel Borches
Laboratory of Design, Production and Management,
Department of Engineering Technology,
University of Twente
P.O.Box 217
7500 AE Enschede
The Netherlands
g.m.bonnema@utwente.nl p.d.borches@ctw.utwente.nl

Abstract. Literature on how the design process should be conducted is abundant. Also literature on how the design process is performed in practice can be found. An issue that has received less attention is how to keep overview while the design project evolves. This article will look at several aspects of overview in the design process. Encapsulation is proposed as a means to avoid saturation of system designers, yet providing enough detail information. The notion of sampling the design process is introduced, as an equivalent to sampling a signal in time.

The paper proposes to use the Function - Key drivers architecting tool (FunKey) as a means to keep overview. Using this tool, it is possible to encapsulate details, yet provide insight and overview where needed. Also, it is possible to sample progress in the design project at a much higher frequency than using the (document) reviews described in system engineering guidelines like the INCOSE handbook.

1 Introduction

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 is that these people are increasingly more distributed over the planet.

The paper 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. An example on a somewhat simpler system, a personal urban transporter (PUT), is used.

2 The problem

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

Close Next

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 workflow 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 of 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, Lutters et al. 2007).

(Axelsson 2002) contends that tools and methods do not design systems; humans do. Although there have been attempts for 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 order that information, he is able to make better informed decisions and use his creativity more efficiently, resulting in better products.

2.1 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 to 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 center 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" airplanes. (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 NRC 2007; van der Heide 2007a; van der Heide 2007b). The cause of the delays were not in the difficult items, but in by themselves simple parts: the wiring for the Airbus A380. In a comparable system, the Boeing 787 Dreamliner, bolts and nuts caused delays (van der Heide 2007b). 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.

2.2 Observed trends in design of complex systems

Given the fact that in both cases the products to be designed are complex, business to business products (medical imaging systems and commercial aircraft), and 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 maybe 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 the preparation thereof. However, the costs of the product are largely *defined* in the early phases of design: system and concept design, see (Kals, Luttervelt et al. 1996 p.17; Zuo and Director 2000).

The processes that have been used in creating the older products will be stressed enormously because of these developments when the required results are the present day products with the corresponding tighter expectations and requirements. In particular the way development teams used to be organized, in disciplines, hampers the development of products that require the synergetic 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 synergetic way of working, however. 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 over the current state of the design. We will present a simple tool that can be used for this.

2.3 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),

Prev First Next

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 paper emerges from research on systems design and conceptual design. In this track, it has been researched how concept and system designers can be supported (Bonnema 2008).

The present paper emerges from the Darwin project (Laar, America et al. 2007). This applied research project is currently conducted at Philips Medical Systems and focuses on the evolvability of software-intensive systems, with as use case Magnetic Resonance Imaging (MRI) systems. Five universities cooperate with Philips Medical Systems and the Embedded Systems Institute to develop understanding and methods to support the in-designing of evolvability in complex systems development.

The central issue of this paper is how can the fit between parts that are being designed by multidisciplinary teams, be guaranteed during detailing phases. While on the same time keeping the focus on the system level properties and functionalities.

In Figure 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.

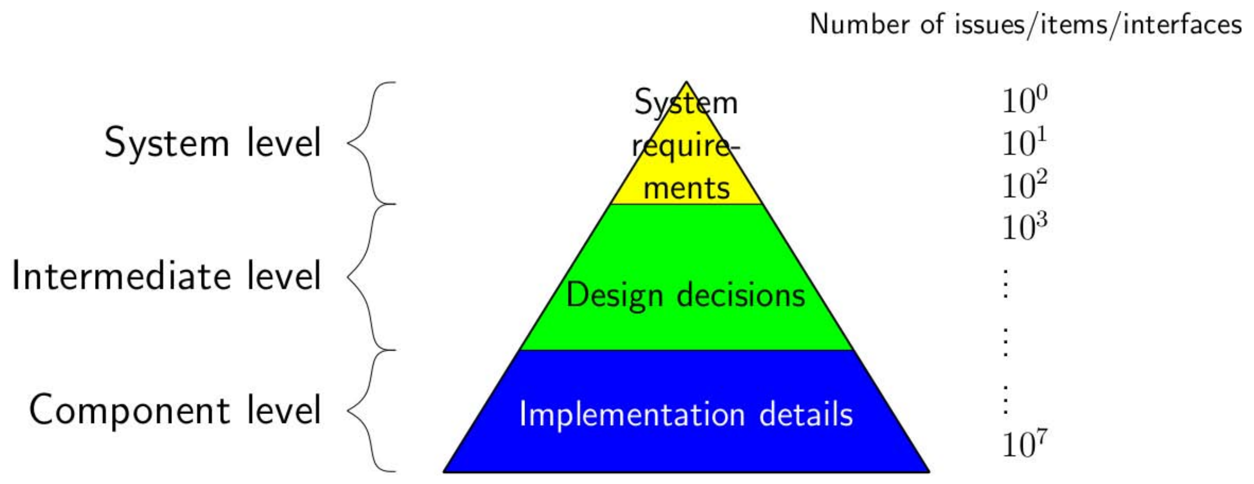

Figure 1: Pyramid showing the increase of issues when moving in the direction of detail design. After (Muller 2007).

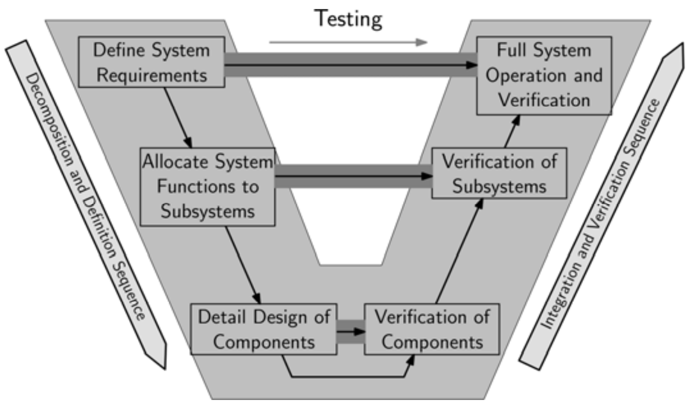
Testing
Define System Requirements
Full System Operation and Verification
Allocate System Functions to Subsystems
Verification of Subsystems
Detail Design of Components
Verification of Components
Decomposition and Definition Sequence
Integration and Verification Sequence

Figure 2: The "Vee" Process Model after (Blanchard and Fabrycky 1998).

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, and that runs 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.

3 Overview in the design process

Overview means a broad, comprehensive view. However, we are confronted with the fact that humans can only handle a limited number of inputs; *the magical 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, 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 come 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.

The well-known "Vee" process model (Blanchard and Fabrycky 1998) shows a process to achieve this decomposition, see Figure 2. The system is parted into subsystems and subsequently components. For each level requirements are compiled, that are translated into test specifications. In the left branch of the Vee, the amount of detail increases and so does the number of components. Referring to Figure 1, we move downwards in the pyramid. In the right branch of the Vee, the components are tested according to their test specifications, and integrated with other components to form subsystems. Next the subsystems are integrated into a full system.

The result of using this Vee process model is that on a higher level in the hierarchy details will be at first unknown, and later masked, in order to not saturate the designers at that level (the system engineers). They have to handle information on more of the

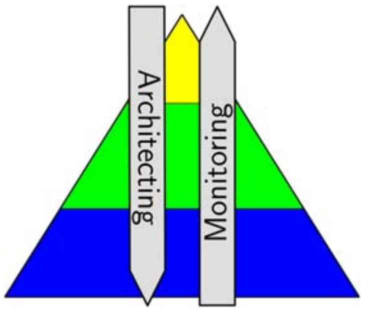

Figure 3: Architecting and monitoring in the pyramid of Figure 1.

Prev First Next

system, and thus because of their limited, but probably large, information management capacities, the information for each part will contain less detail.

The goal should be that *only the relevant information* needs to be handled by the system engineers. The resulting question then is: how can we determine what is relevant information? If we can design a support tool that shows relevant information on each level of the process, without saturating the information handling capacity of the user at each level, we can truly design with overview.

Next we will look at two approaches that can be used to achieve this goal: "encapsulation" and a combination of "discretisation" and "oversampling".

4 Encapsulation as a means for overview

Encapsulation is a term from software engineering, in particular from the object oriented method (Mattos, Meyer-Wegener 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 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). 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.

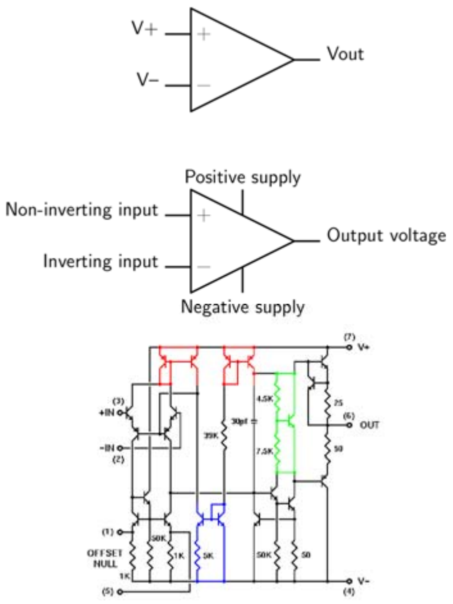

Figure 4: Encapsulation of an OpAmp. The top image shows a fully encapsulated scheme ("black box"), the middle image exposes some of the nonideal behaviour ("grey box"), while the bottom image shows the internals of an OpAmp.

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. 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 thus worthwhile to have a tool that on the one side helps in creating the partitioning of the system into black boxes (the architecture, (Bonnema 2006a)), while on the other hand it channels information from the detail hierarchical levels (the bottom of the pyramid in Figure 1) to the system level (the top of the pyramid), see Figure 3. In Section 6 such a tool is presented.

Prev First Next

5 Discretisation and sampling in the design process

In many cases, there are only a limited number of moments at which the design process is reviewed, mostly by reviewing a document, see Figure 5. 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 2), 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 samples in Figure 5, representatives of adjacent modules, or the next higher hierarchical level may deduct the course marked as "Extrapolated course", whereas in reality the course is drastically adjusted shortly after the second sample moment. Using terms from control engineering, we can state that due to the discretisation of the design process by a documentation system, the process is *undersampled*. This will lead to erroneous results, just as undersampling a signal will prevent proper reconstruction.

In (Muller 2004a), it is proposed to use WWHWWW questions to sample the design process (why, what, how, who, when, where) by applying the following simple rule:

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

This sampling has to be done by one or more system engineers or system architects.
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

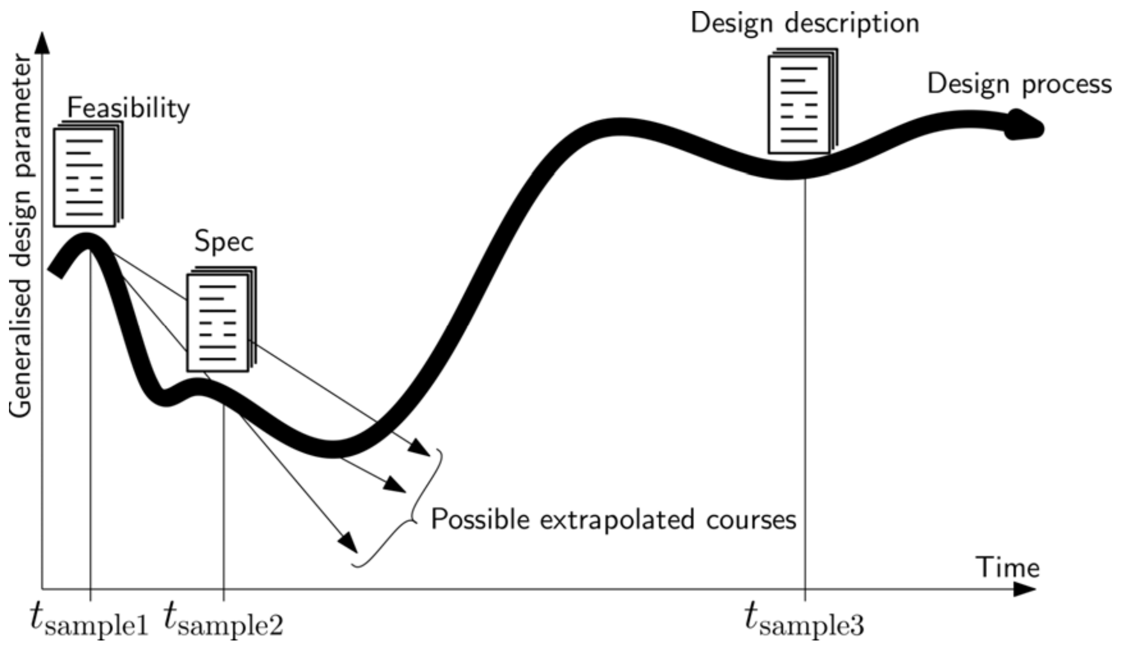

Figure 5: The course in a design project, showing three "sample-moments": Feasibility study, Specification and Design description.

other areas like mechanical or mechatronic engineering, such a mechanism is not present. We will present in section 6 a method that can be used by the system designers to track progress on any hierarchical level.

There is a difference between extracting the system's architecture from an already present design, and aiding the design process in such a way that the architecture is documented while it is created. The former, which will be published about shortly by the same authors, is in fact *reconstructing* the mid section of the pyramid in Figure 1 from information contained in the detail designs and system level specifications, whereas the latter tracks any progress in working from top to bottom and vice versa. In the next section we will present how the FunKey method can be used in this second case.

6 FunKey architecting tool for overview

The FunKey architecting approach has been presented before (Bonnema 2006a; Bonnema 2008). It bases on the identification of functions and key drivers, and uses system budgets to create overview over the system and to create architectures. Moreover, TRIZ (Altshuller 1997; Salamatov 1999) is easily applied from the FunKey method, to simplify the architecture and to find new solutions for implementing the system (Bonnema 2006b). Below we will explain the FunKey method in short, and treat how it can be used for oversampling the design process.

The FunKey method uses a coupling matrix C to connect functions to key drivers. Functions are tasks to be performed by the system. In (Blanchard and Fabrycky 1998) a function is defined as "a specific or discrete action that is necessary to achieve a given objective" (p.62). In practice we can describe a function as a noun with a verb, like *expose wafer*, *transport sand*, *create image*. In general a function can be split into several sub-functions. Transport sand, for instance, contains (among others) the subfunctions *contain sand*, *accelerate sand*, *decelerate sand*, *load sand*, and *unload sand*. It is seen that functions and function models are important in the early phase of the design process (Bonnema and van Houten 2006).

Key drivers are generalised requirements that express the customers' interest (Muller 2004a). Where 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 kilometer* for a truck.

Using the functions and key drivers, the FunKey architecting procedure is as follows (see Figure 6).

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

System budgets, in this context, are ways of distributing a system requirement over the parts constituting the system. Examples are the distribution of available power over the electronics and the distribution of the allowed positioning error over the subsystems. Budgets are often used in developing high-tech systems (Frericks, Heemels et al. 2006). They provide guidance in defining sub-system requirements. We will show later that the structure of the budgets can also be used for sampling the process, and for applying encapsulation.

After the initial matrix 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

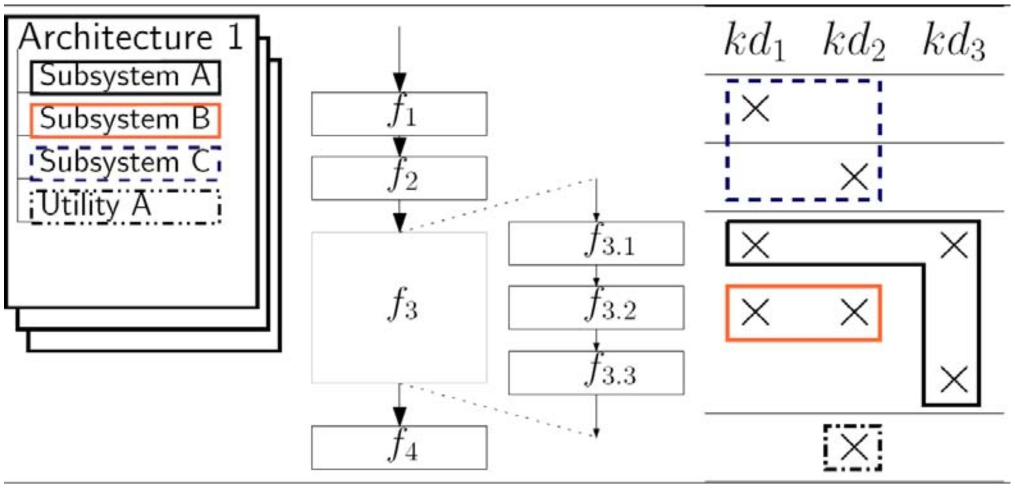

Figure 6: The FunKey architecting method. To the right the coupling matrix C **is shown that connects the functions in the block diagram to the key drivers** kd_i**. On the left, one architecture is shown. The subsystems are marked in the coupling matrix. On the top level, functions can also be assigned to the user, the environment and the supersystem.**

using either numbers, or symmetrical triangular fuzzy numbers (STFN) (Chan and Wu 2005). 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.

The FunKey procedure visualizes 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. For more information, and particularly the relation between the presented method and other methods like Axiomatic Design (Suh 1990) and Quality Function Deployment (QFD), the reader is referred to the earlier mentioned references (Bonnema 2006a; Bonnema 2008).

The procedure above is invented for the *creation* of the architectures, and presented as such in (Bonnema 2006a). Interesting is that the FunKey method can also be used for oversampling the design process, so for the *monitoring* of the design process. Once the initial matrix is filled out, it can be worked out for a next hierarchical level. If on this next level a coupling between a (sub-)function and key driver, or (system) requirement is necessary, this will also modify the top-level matrix. Thus, couplings on the more detailed levels will result in couplings at higher level.

Moreover, as the coupling matrix C is used to create system budgets, these budgets can be used for maintaining overview. As mentioned above, the matrix first is filled with crosses (when a contribution is present). When more information becomes available, the crosses will be replaced by numbers (or, STFNs). The information can follow two routes: *soll*-values that propagate in a top-down manner; and *ist*-values that are the result of detail design work and propagate bottom-up, see Figure 3.

Of course, this does not replace a system engineer who by experience discusses critical points of the design with the designers. Rather, the approach presented above will provide input to the system

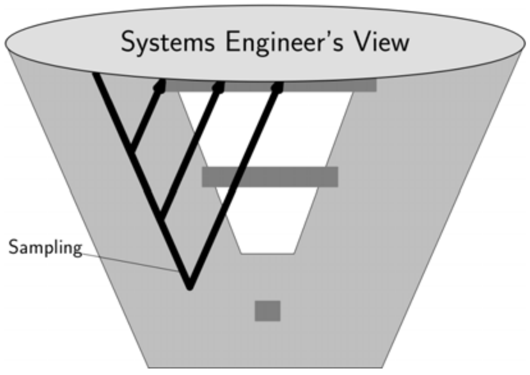

Figure 7: Sampling and the systems engineer's view in the Vee-model from Figure 2.

Prev First Next

Table 1: Top level FunKey diagram for a Personal Urban Transporter. The architecture that is worked out here is marked Arch3. (env.load = environmental load; U = user, PUT = Personal Urban Transporter, Env = Environment).

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

engineer as is shown in Figure 7. The system engineer can also sample the process at other depths, and at the test sequence definition process.

By setting up the system as a database, it is possible to “zoom” in and out. Functions can be expanded into subfunctions and further. Also, the key drivers can be expanded into the system requirements. This way encapsulation (Section 4) is brought to designing multidisciplinary systems.

7 Example

As an example we will (re-)use the Personal Urban Transporter presented in (Bonnema 2006a; Bonnema 2006b). A personal urban transporter, or short PUT, is a simple vehicle for commuting. It should be an environmentally friendly alternative for use in cities, and for not too long distances. The key drivers are: cost per kilometre, environmental load, safety and convenience. The main functions are derived as: drive vehicle, maintain balance (perpendicular to direction of movement), maintain posture (in direction of movement), create light, steer, navigate and support user. In (Bonnema 2006a), three architectures were presented, of which one, shown in the last column of Table 1, will be used in this example. The architecture shows whether the specific function is assigned to the system (PUT), the environment or the user. The crosses show to which key driver each of the functions (or subfunctions) contribute to.

In the next step, the functions that are assigned to the PUT, receive a part of the available budget for the key drivers, and subsequently of the requirements that belong to the key driver. This can initially be in percentages of the available amount, but when the subsystem specifications are compiled, these percentages will be converted. Just as on the top level the functions are assigned to the PUT, the user and the environment, on the next hierarchical level the functions are assigned to the different subsystems. A small part of the resulting (underlying) FunKey table is shown in Table 2. Here several of the functions are expanded into sub-functions, and the key drivers are split into top-level requirements.

Table 2: Part of the detailed FunKey matrix of the Personal Urban Transporter in Table 1. The numbers are allocated percentages of the total.

Function	cost	power	visibility	effort	sub-system
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	35			7	Seat
provide power					Electrical
~ store power	5	1			
~ deliver power	5	1			

This table can be worked into budgets for all the subsystems. These budgets can be presented in a tree-structure, like the power budget tree shown in Figure 8. This has been a very insight-full means of presentation for the overlay budget at ASML (www.asml.com). It has provided detail designers overview and context information (Muller 2004b; Muller 2004a).

The addition we like to make in this paper is to explicitly add the bottom-up route in the budgets. In a table like the one in Table 2, an extra column to accommodate for the 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, 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

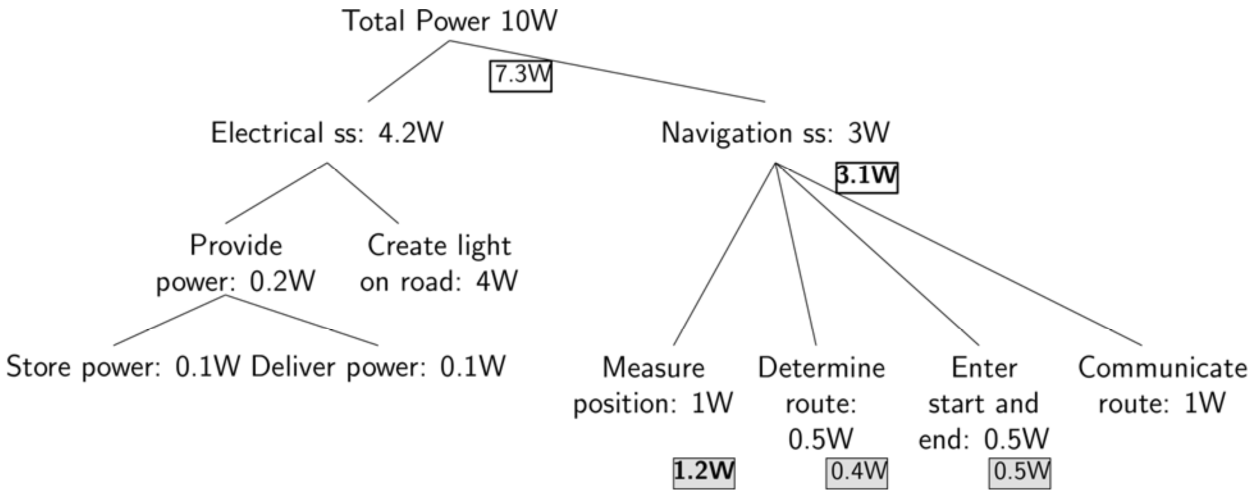

Figure 8: Budget tree for the PUT in Table 1 and Table 2. Only the power tree is shown. Similar trees can be produced for cost, visibility and effort. The numbers in grey boxes show the values entered by the designers. The numbers in white boxes are calculated from lower level values. (ss= Sub system)

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, 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 two 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.

The use of STFN's (Chan and Wu 2005; Bonnema 2008) 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 (Bonnema 2008). Using the navigation subsystem of the PUT of Figure 8, but now representing the expected outcomes of the detail process in STFNs, gives the budget tree in Figure 9. The total power *consumption of the navigation subsystem is thus [2.6, 5.5]W. The membership function is also* shown in Figure 9, 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.

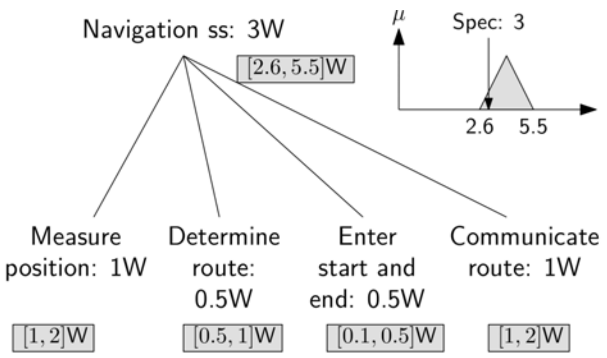

Figure 9: Power budget tree for the Navigation subsystem of the Personal Urban Transporter. The expected outcomes are represented with symmetrical triangular fuzzy numbers (STFNs).

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 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 as overlay in 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.

In fact the first author, when working at ASML, used this method in paper form. As a system engineer he discussed the overlay budget 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.

8 Conclusions and Recommendations

8.1 Conclusions

In order to improve the way present day complex products are designed, it is necessary to provide the system designers with overview of the state of the detail designs. From illustrations in real life, it is concluded that the interconnectivity between subsystems or parts and the distribution of designers over the globe can cause problems.

We have shown that the route from system requirements via subsystem requirements to component requirements is easily followed. However, there should be an accompanying regular stream of information in the opposite direction: from detail design to subsystem design to system design. This stream of information can be used by the system engineers to monitor the process, and verify the decisions against the system requirements and customers' key drivers. At present this stream of information is created by document reviews and the system engineers sampling the process by walking around and asking WWHWWW-questions. These questions are needed because otherwise the design process is sampled only at document reviews, like the module specification review. As these sample moments are scarce, the process is undersampled, leading to erroneous information in the company. The FunKey procedure can be used to sample the process at a higher sampling rate. As this procedure is used to create the system architecture (the top-down route), the framework for propagating information from bottom-up is already in place: system budgets are used in conjunction with the coupling matrix *C*.

The system designer has limited information processing capabilities. In general the limits are higher than for detail designers, but there are limits. As we have seen, the number of information items on the detail design level, over the entire design project, are huge. As the systems designer has to avoid saturation, the number of information items has to be limited. We showed that the principle from software engineering called encapsulation, may be an interesting mechanism to accomplish this. Again, the FunKey tool provides for implementation of encapsulation by expanding or collapsing the functions, subfunctions, subsubfunctions, etc., and expanding or collapsing the key drivers, system requirements, subsystem requirements etc.

8.2 Recommendations and future work

Although the FunKey approach has been applied in practice for the top-down route(Bonnema 2008), the procedure has not been well verified for the bottom-up route, apart from experience with a paper-implementation, several years ago. It is needed to investigate this further. A prototype implementation of the FunKey tool is required for this.

In the near future we will create such a prototype and verify whether the FunKey approach can be used for monitoring the design process.

9 References

Altshuller, G. S. (1997). 40 Principles - TRIZ Keys to Technical Innovation. Worcester, MA, Technical Innovation Center.

Axelsson, J. (2002). Towards an Improved Understanding of Humans as the Components that Implement Systems Engineering. 12th International Symposium of the International Council on Systems Engineering. Las Vegas, INCOSE.

Blanchard, B. S. and W. J. Fabrycky (1998). Systems Engineering and Analysis. Upper Saddle River, New Jersey, Prentice Hall.

Bonnema, G. M. (2006a). Function and budget based system architecting. TMCE 2006. Ljubljana, Slovenia**:** 1306--1318.

Bonnema, G. M. (2006b). TRIZ for Systems Architecting. TRIZ Future 2006. Kortrijk, Belgium**:** I: 87--92.

Bonnema, G. M. (2008). FunKey Architecting - An Integrated Approach to System Architecting Using Functions, Key Drivers and System Budgets. Engineering Technology. Enschede, University of Twente. **PhD**.

Bonnema, G. M. and F. J. A. M. van Houten (2006). "Use of Models in Conceptual Design." Journal of Engineering Design **17**(6): 549--562.

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 **7**(1): 25--34.

Chan, L.-K. and M.-L. Wu (2005). "A systematic approach to quality function deployment with a full illustrative example." Omega **33**(2): 119--139.

Eising, F. (2007). Als je het begrijpt kun je het veranderen. Enschede, Universiteit Twente.

Frericks, H. J. M., W. P. M. H. Heemels, et al. (2006). Budget-based design. Boderc: Model-based design of high-tech systems. W. P. M. H. Heemels and G. J. Muller. Eindhoven, Embedded Systems Institute**:** 59--76.

INCOSE SEH Working Group (2000). Systems Engineering Handbook, INCOSE.

Kals, H. J. J., C. A. v. Luttervelt, et al., Eds. (1996). Industriële Productie. Middelburg, De Vey Mestdagh.

Laar, P. v. d., P. America, et al. (2007). The Darwin Project: Evolvability of Software Intensive Systems. Third International IEEE Workshop on Software Evolvability (2007). Paris.

Lewes, G. H. (1875). The Problems of Life and Mind - First Series: The Foundations of a Creed. London, Kessinger Publishing (2004).

Lutters-Weustink, I. F., D. Lutters, et al. (2007). Domain integration by means of features. International Conference on Competitive Manufacturing; COMA'07. Stellenbosch.

Mattos, N. M., K. Meyer-Wegener, et al. (1993). "Grand tour of concepts for object-orientation from a database point of view." Data & Knowledge Engineering **9**(3): 321--352.

Miller, G. A. (1956). "The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information." Psychological Review **63**: 81-97.

Muller, G. J. (2004a). CAFCR: A Multi-view Method for Embedded Systems Architecting. Delft, Delft University of Technology.

Muller, G. J. (2004b). "The Waferstepper Challenge: Innovation and Reliability despite Complexity." 2005, from http://www.gaudisite.nl/IRCwaferstepperPaper.pdf.

NRC (2007). Eerste Airbus A380 geleverd, anderhalf jaar te laat. NRC Handelsblad. Rotterdam. **15 oktober 2007:** 11.

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 **9**(2): 107-119.

Salamatov, Y. P. (1999). TRIZ: the right solution at the right time: a guide to innovative problem solving. Hattem, The Netherlands, Insytec.

Schulz, A. P. and E. Fricke (1999). Incorporating flexibility, agility, robustness, and adaptability within the design of integrated systems - key to success? Digital Avionics Systems Conference, 1999. Proceedings. 18th.

Suh, N. P. (1990). The Principles of Design. New York, Oxford, Oxford University Press.

van der Heide, L. (2007a). A380 van Singapore Airlines blijft de enige dit jaar. NRC Handelsblad. Rotterdam. **16 oktober 2007**.

van der Heide, L. (2007b). 'Vliegtuig in elkaar zetten blijkt lastig'. NRC Handelsblad. Rotterdam. **26 oktober 2007**.

Zuo, J. and S. W. Director (2000). An Integrated Design Environment for Early Stage Conceptual Design. Design, Automation and Test in Europe Conference and Exhibition 2000. Proceedings.

Zwikker, R. (2007). Organisation of project teams at Demcon; personal communications**:** Organisation of project teams at Demcon: multidisciplinary.

This work has been carried out as a part of the DARWIN project at Philips Medical Systems under the responsibilities of the Embedded Systems Institute. This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

Biography

G. Maarten Bonnema is an assistant professor at the Laboratory of Design, Production and Management of the Faculty of Engineering Technology at the University of Twente. He studied Electronic Engineering, and did a two year designer course on Technical Systems. He has worked as a Systems Engineer at ASML. In 2006 and 2007 he was involved in the design of a wafer stepper at MAPPER lithography (part time). His research involves supporting system designers, conceptual design and mechatronic design. He teaches design in general, systems engineering and is a tutor in various student projects.

P. Daniel Borches received his Master's degree in Telecommunication Engineering from the University of Madrid Carlos III in 2004. He has worked in companies such as Nokia and Telefonica R&D. From 2006 he is working at Philips Medical Systems Netherlands as a researcher while doing his Ph.D. at the University of Twente. The main focus of his research is Systems Engineering and Systems Architecting applied in the industrial sector.