The Engineers' Innovation Toolkit

G. Maarten Bonnema[1]

[1]University of Twente, Department of Engineering Technology, Laboratory of Design, Production and Management, The Netherlands

Abstract

Most engineers nowadays receive a mono-disciplinary education: Mechanical engineering, Electrical engineering etc. Contradictory, the products they have to design are ever more multidisciplinary and integrated. This requires a different mindset. This paper discusses four tools that fit in the engineers' toolkit to approach these multidisciplinary problems: TRIZ, Systematic Inventive Thinking, Quality Function Deployment and FunKey Architecting. The tools are discussed and rated on four scales: difficulty of problems, complexity of problems, design phase and learning effort. From the characterization a set of heuristics is derived that help in choosing the appropriate tool from the toolkit.

It is concluded that the four tools largely complement each other and should therefore be part of every engineer's toolkit.

Keywords

TRIZ, SIT, QFD, FunKey, Toolkit, Selection, Heuristics, Complex system design

1 INTRODUCTION

Designing present day products has become a complicated process. Due to the highly integrated nature of many modern products, they contain state of the art mechanics, electronics, optics and software, and due to the minimized time to market, designers of these products have to process large amounts of diverse information in a short time span. Successful products require a sufficient mix of newness and familiarity.

Innovation is the successful development and deployment of a new activity. Therefore to innovate, engineers need to develop new activities, overcome or circumnavigate the problems associated with the design, and develop a product that is easy to produce. The reuse of previous developments should be promoted. These three types of activities are very diverse.

The paper will look at how innovation can be made readily accessible for engineers. For this we will first characterize four innovation approaches: TRIZ, Systematic Inventive Thinking (SIT), Quality Function Deployment (QFD) and FunKey Architecting. The latter, though earlier presented at a TRIZ Future Conference [2], is assumed to be less well-known. It will be treated more extensively than the other ones.

Based on the descriptions, a comparison can be made. We will derive a set of heuristics that help in deciding which tool to use, given the problem at hand. We will see that to develop new product ideas with a sufficient portion of newness can be stimulated with SIT. TRIZ can be used to overcome the many problems of implementing the design. FunKey can be used to guide and monitor the entire design process, and to stimulate reuse of previous developments. Largely the same holds for QFD.

2 DESIGNING COMPLEX SYSTEMS

Product design is getting more complicated at present. This is due to the increased number of product functions and due to the increased difficulty of these functions. Also, the number of people involved has increased [3] while simultaneously their distribution over the planet has increased. This reflects the history of the organisations and the increased pressure on costs, and is facilitated by the present communication means.

The tools used are not always up to these new working environments [4]. In that reference it is contended that in the "drawing board environment", the workflow was from "big picture" to detail. The big picture used to be always present in the form of the large overview drawings. 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.

We like to emphasise that systems are designed by humans, not by tools [5]. Although there have been attempts for automated design systems, we focus on helping the designer. In this process we will not take the innovative steps from the human responsibility.

Figure 1 illustrates the problem. At the top of the pyramid there are only a few requirements from the stakeholders.

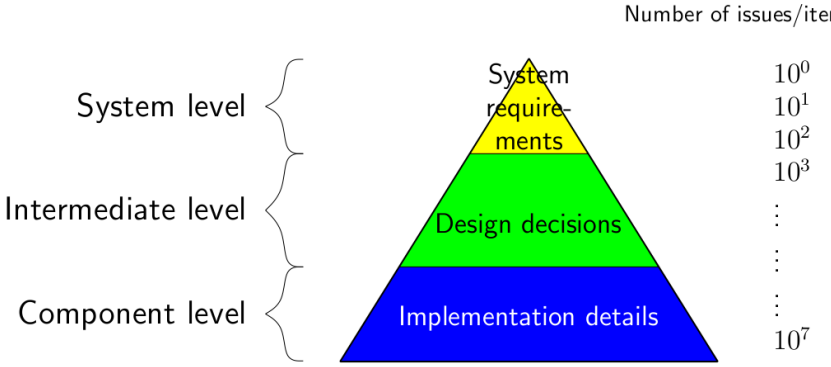

Figure 1: Pyramid of the number of issues, items and interfaces occurring in different stages of the design process [1].

These are at a high level of abstraction. The requirements should reflect 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. The complexity at the bottom of the pyramid is high. Fortunately this complexity can mostly be split into mono disciplinary blocks. This represents the architecture of the system. 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. Although, as is shown in [6] minimizing the coupling should be aimed at, total decoupling is impossible as a system consists of cooperating parts. 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 [7, 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 reductionist approach), but it emerges [8] from the cooperation of and interaction between the subsystems.

At the top of the pyramid, the system level, the number of issues is relatively limited and can be dealt with by a few persons.

This paper will look at how TRIZ-related approaches/ methods can be used in the problems described in this pyramid of designing complex systems. Although many flavours of TRIZ-related tools exist, we will look at the following:

- TRIZ;
- Systematic Inventive Thinking (SIT) and Advanced SIT (ASIT) [9-11], (www.start2think.com and www.sitsite.com);
- FunKey Architecting [2, 12, 13], and
- Quality Function Deployment (QFD) [14, 15].

Each of these approaches will be characterised in the next section.

3 CHARACTERIZING THE APPROACHES

Making a taxonomy of design tools, methods and approaches has been done before ([16, 17] and others). For classifying the different approaches, some metrics are needed. Please note that it is not required to have an exact and absolute measure. It suffices to use scales of comparison. Therefore we will not use any units on the scales.

The first metric we use is the difficulty of the problem at hand. Easy problems relate to routine design work. Difficult problems are those that require a new approach, a new technology. For this we can use the TRIZ five levels of innovation [18]. In this paper we will denote level one as "easy"; level four as "difficult". Level five is left out of consideration. Discovering new phenomena is not applicable in design. (Fundamental) science is needed for that.

In designing complex systems, the complexity of the problem at hand is an appropriate measure. In this sense, aggregate complexity [19] is meant, where individual elements work in concert to create systems with behaviour that has emergent characteristics [8]. In [13] the complexity scale ranges from simplex to composite. This is what we will use in this paper as well.

A third characteristic is the phase of the design process where the approach is applicable. Although there are many design process models described (see among others [16, 17]), the *communis opinio* is that the following phases occur:

1. Establishing the need;
2. Requirements and specification development;
3. Conceptual design;
4. Embodiment design;
5. Detail design.

These five phases will be used.

Finally, we have to look at the effort required to master the method at hand. We will use a one to five scale again, where one represents "easy". So within half a day proficiency can be reached with the method. Five represents "laborious". This means years of practice and training are required for proficiency.

Combining these four scales in one diagram, results in the radar plot in Figure 2. Please note that because we use four characteristics, the diagram looks like a quadrant plot. But each of the four axes should be regarded independently.

In the following subsections, four shapes will be shown in this plot; each representing one of the design approaches.

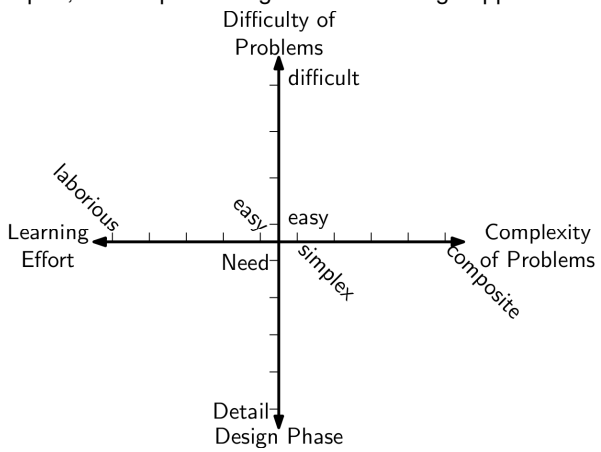

Figure 2: Radar plot used for characterizing the design approaches.

3.1 TRIZ

The Theory of Inventive Problem Solving (TRIZ) requires the shortest introduction in the context of this Conference of TRIZ experts from around the world.

TRIZ is particularly useful in solving difficult problems. As Altshuller has concluded, using TRIZ can help in achieving level 2-4 innovations instead of mere improvements at level 1.

TRIZ is used to resolve contradictions in one of several ways. These contradictions can be (very) hard to remove, yet there are in each case *single* contradictions to resolve. TRIZ is not (by itself) apt for solving (aggregate) composite problems.

As for the phase of the design process, TRIZ (except maybe for the trends of evolution) cannot be used to identify a need or compile the requirements. Finally, becoming a TRIZ-expert requires long training and lots of practice.

Combining these observations into the radar plot defined in Figure 2 results in Figure 3.

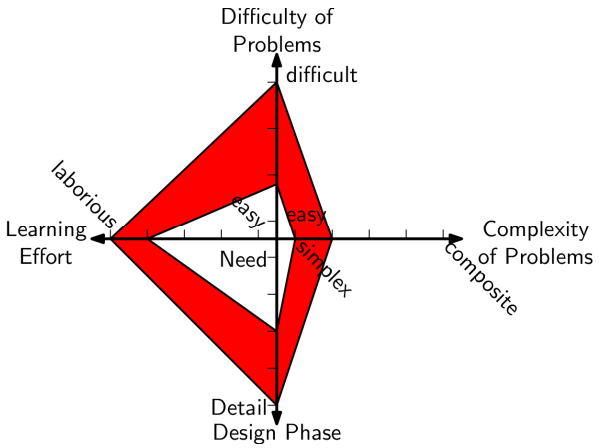

Figure 3: TRIZ characterized in the radar plot of Figure 2.

3.2 SIT

Like TRIZ, (Advanced) Systematic Inventive Thinking (SIT) uses a set of rules or tools to solve problems. Yet a large difference is that SIT helps in particular in finding new product ideas. It is less apt for the resolution of problems and contradictions while designing. Therefore, SIT is used in the earlier phases of the design process. Also, as said, SIT is not particularly fit for solving problems. It merely tries to find alternative solutions, without thoroughly analysing the problem. Although examples exist of SIT as problem solving (the antenna example [10], for instance), it works best in creating new product ideas and concepts.

Finding new products involves many aspects to consider: the customer, the market, the technology available etc. To find suitable product ideas is considered a moderately complex problem. Therefore, the simplex-composite characteristic for SIT is somewhat higher than for TRIZ. Please note that SIT can also be used for creating advertisement campaigns.

Learning SIT requires far less effort than TRIZ. This is an inherent characteristic of the method. It uses a very limited set of tools and principles.

Figure 4 shows the characterization of SIT.

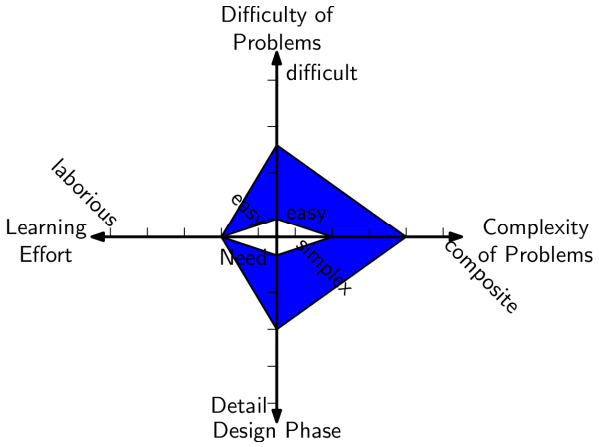

Figure 4: Systematic Inventive Thinking characterized in the radar plot of Figure 2.

3.3 FunKey Architecting

FunKey Architecting [2, 12, 13] requires a bit more of an introduction. It is based on the identification of functions and key drivers. Overview over the system is created and maintained with the help of system budgets. Using these, system architectures can be created. Moreover, TRIZ is easily applied from the FunKey method. It can help in simplifying the architecture and in finding new solutions for implementing the system [2].

The FunKey method uses a coupling matrix *C* to connect functions to key drivers. Functions, as known in TRIZ, are tasks to be performed by the system. [7] defines a function as “a specific or discrete action that is necessary to achieve a given objective” (p.62). Practically, a function is described with a noun and a verb. Examples are *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*. Functions and function models are important in the early phase of the design process [20, 21].

Key drivers are generalised requirements that express the customers’ interest [22]. 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 5).

1. Identify functions and 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.

With system budgets a system requirement can be distributed over the parts constituting the system, as defined by the architecture. Examples are the distribution of available power over the electronics and the distribution of the available time over the (sub) processes in a real time computer program. Budgets are often used in developing high-tech systems [23].

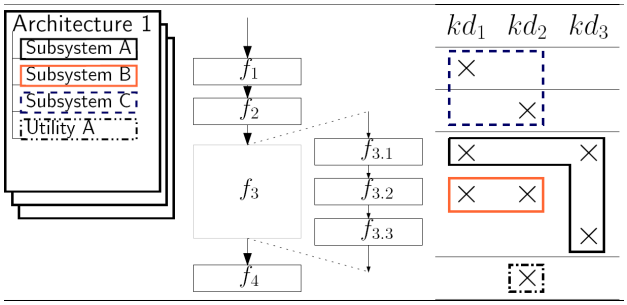

Figure 5: The FunKey architecting method. To the right the coupling matrix ***C*** is shown that connects the functions in the block diagram to the key drivers $\boldsymbol{kd_i}$. On the left, one architecture is shown. The subsystems are marked in the coupling matrix. On the top level, functions can also be assigned to the user, the environment and the supersystem.

After initially the matrix *C* has been filled with crosses or ones (when there is a contribution from the corresponding function to the key driver), the contributions can be quantified using either numbers, or symmetrical triangular fuzzy numbers (STFN) [24]. 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.

FunKey Architecting helps in visualizing implicit architectural decisions. Therefore, it is valuable for a team of architects and for communicating architectural decisions between architect and specialist and/or detail

designer. For more information, the reader is referred to the earlier mentioned references [12, 13].

The procedure above is invented for the *creation* of the architectures, and presented as such in [12]. Interesting is that the FunKey method can also be used for *monitoring* the design process [25]. 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. 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.

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.

From the short introduction to FunKey Architecting above, it can be easily concluded that the whole range of easy to difficult and simplex to complex problems can be handled by the FunKey approach. Moreover, it is wise to use it throughout the design process, from establishing the need to the detail design phase. This way, the system designers can maintain overview, while the detail designers get sufficient context information.

Learning FunKey is not difficult, assuming one has some experience in designing complex systems. In short, FunKey architecting can be classified with Figure 6.

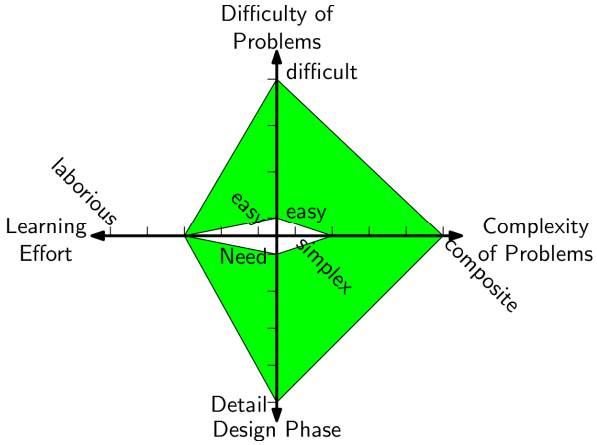

Figure 6: FunKey Architecting characterized in the radar plot of Figure 2.

3.4 QFD

The fourth and last approach is Quality Function Deployment. This method originates from Japan in the 1960's, and was introduced by professors Shigeru Mizuno and Yoji Akao [14]. Its aim is to incorporate quality in every product, from the outset of the design project. There are several attempts of joining TRIZ and QFD like [26, 27]. QFD is widely used. Therefore the way of working for QFD is assumed to be familiar. The main concept of QFD is that the need and the requirements of the customer are investigated and well documented. As mentioned on [14], QFD:

1. Seeks outspoken and unspoken customer needs from fuzzy Voice of the Customer verbatim;
2. Uncovers "positive" quality that wows the customer;
3. Translates these into designs characteristics and deliverable actions; and
4. Builds and delivers a quality product or service by focusing the various business functions toward achieving a common goal—customer satisfaction.

As with FunKey, QFD can be used in the entire spectrum from easy to difficult and simplex to composite problems. Yet, whether its application in easy and simplex cases is useful is questionable. Further, QFD is particularly aimed at defining the customer's requirements so that they express the need. In the embodiment and detail phase, these requirements can be used as a reference but there is no feedback mechanism.

In general, QFD is not thought to be difficult or time consuming to learn. Thus QFD can be characterized as shown in Figure 7.

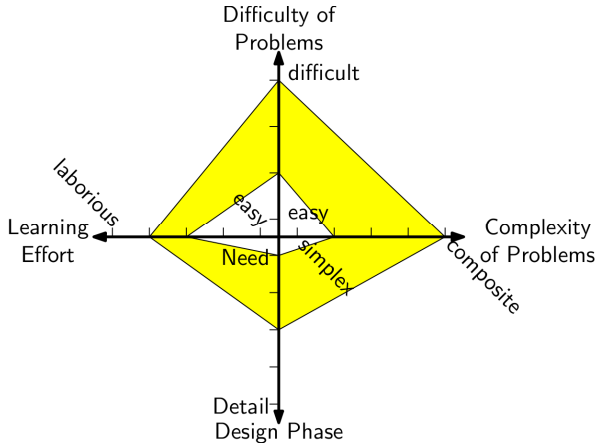

Figure 7: QFD characterized in the radar plot of Figure 2.

4 DISCUSSION

Having characterized these four approaches and restating that we look at their applicability in designing complex systems as shown in Figure 1, it is worthwhile to discuss which approach(es) is (are) best applied in the different parts of the pyramid.

We start at the top of the pyramid where the system requirements have to be compiled. This requires insight in the need, either outspoken or latent. The first situation is where QFD can be used very well. The latent need of a customer can be found by finding new product ideas that might fit that latent need. When such a product is thought up and shown to the customer, the need may become explicit. This results in the first heuristics, valid at the top of the pyramid:

1. For outspoken customer needs use QFD.
2. To investigate any latent needs use SIT.

When the product ideas created with SIT are discussed with the customer, these ideas may be further analyzed using QFD, of course:

3. The need of the customer should be analyzed using QFD.

Then let us move to the bottom of the pyramid. Here many issues are to be resolved. There will be many contradictions to solve, most of them within one domain, or at most between two domains. This is where TRIZ is at its best. When the designers are trained in applying TRIZ, the work at the bottom of the pyramid may progress more rapidly and better directed than without TRIZ.

4. TRIZ should be used for single- and two-domain problems.

Finding these kinds of problems is not always easy. This is where FunKey (and also QFD) comes in. As shown in [2, 13], the FunKey approach can quickly show contradictions to be solved. Moreover, using the priority matrices introduced in [28] and extended in [13], there is no need to describe the contradictions. The possible

areas of improvement can be shown by connecting the key drivers with the 39 TRIZ parameters.

5. Use the FunKey approach to investigate areas for improvement early in the design process.
6. Use FunKey to establish contradictions to be solved with the TRIZ parameters.

The two heuristics mentioned above also hold for using QFD [26].

The original goal of the FunKey approach was to assist the system architect; the person defining the system's basic structure and allocation of functions. It is shown [13] that FunKey can be used for that very well. Thus FunKey should be used to divide the system design in smaller, more manageable pieces; the modules or subsystems. The overall performance can still be monitored using FunKey:

7. Create the system architecture using FunKey architecting.

During the design process of a complex system, several people have to monitor the progress, and ensure the basic concepts are not violated [25]. This monitoring can be done easily using the system budgets created with the FunKey approach; for instance by using STFN's as described above.

8. Track progress, in particular from a technical point of view, using FunKey.

We have not yet looked at the learning effort for the four methods. From the figures shown above, we can see that TRIZ requires most learning and practicing effort. It is therefore unfortunate that TRIZ has its largest effectiveness at the more detailed levels of the design process. There, the number of people working is the largest. This disadvantage can be overcome, as suggested more often, by instructing all engineers in the basic principles of TRIZ so that they can identify the problems where TRIZ is expected to be effective. In cooperation with a few TRIZ-experts in a team, these problems can be solved. (This solution is an illustration of the segmentation principle.)

As the other approaches do not require a large learning investment, most people in the team can learn them.

From these heuristics, we can conclude that the four approaches treated in this paper largely complement each other. There is some overlap between QFD and FunKey, but this may work for the better when a possible connection between the two approaches is investigated further.

Therefore, referring to the title of the paper, four tools that belong in the engineers' innovation toolbox are TRIZ, SIT, FunKey and QFD. It is not required for each engineer to be proficient in all four of the methods, yet a basic understanding is welcome. A few TRIZ-experts are needed in each design team.

5 CONCLUSIONS

A summary of the above is shown in Figure 8. SIT is particularly suitable in the top of the design pyramid. It loses its use when more details have to be considered. TRIZ is appropriate in the base of the pyramid and in particular for solving the many difficult problems associated with designing high-tech systems.

FunKey and QFD are both useful in dissecting the design problem into smaller pieces. In addition, FunKey is very useful in creating the architecture and for tracking progress in larger projects.

6 RECOMMENDATIONS

It can be concluded that there are similarities between QFD and FunKey. Although their goal differs, it is wise to investigate a better connection between the two.

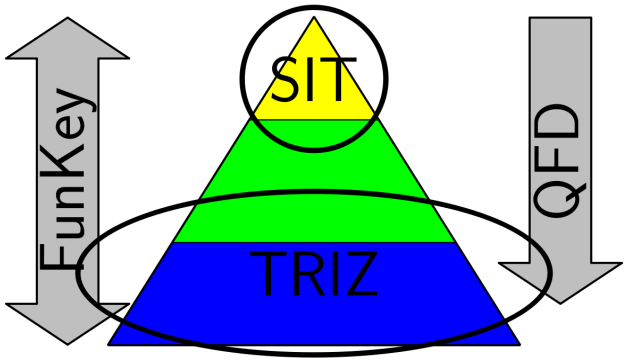

Figure 8: The four approaches and the pyramid of system design.

REFERENCES

[1] Muller, G.J. *Design objectives and design understandability*. 2007 [Last access 7 june 2007]; Available from: http://www.gaudisite.nl/DesignUnderstandabilitySlides.pdf.

[2] Bonnema, G.M., *TRIZ for Systems Architecting*, in *TRIZ Future 2006*. 2006: Kortrijk, Belgium. p. I: 87--92.

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

[4] Ottoson, S., *Qualified Product Concept Design Needs a Proper Combination of Pencil-aided Design and Model-aided Design Before Product Data Management*. Journal of Engineering Design 1998. 9(2): p. 107-119.

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

[6] Suh, N.P., *The Principles of Design*. Oxford Series on Advanced Manufacturing, ed. J.R. Crookall and M.C. Shaw. 1990, New York, Oxford: Oxford University Press.

[7] Blanchard, B.S. and W.J. Fabrycky, *Systems Engineering and Analysis*. Third ed. 1998, Upper Saddle River, New Jersey: Prentice Hall.

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

[9] Goldenberg, J., et al., *Finding Your Innovation Sweet Spot*, in *Harvard Business Review*. 2003, Harvard Business School Publishing Corporation.

[10] Goldenberg, J. and D. Mazursky, *Creativity in product innovation*. 2002, Cambridge: Cambridge University Press.

[11] Stern, Y., I. Biton, and Z.e. Ma'or, *Systematically Creating Coincidental Product Evolution Case Studies of the Application of the Systematic Inventive Thinking (SIT) Method in the Chemical Industry*. Journal of Business Chemistry, 2006. 3(1): p. 13--21.

[12] Bonnema, G.M., *Function and budget based system architecting*, in *TMCE 2006*. 2006: Ljubljana, Slovenia. p. 1306--1318.

[13] Bonnema, G.M., *FunKey Architecting - An Integrated Approach to System Architecting Using Functions, Key Drivers and System Budgets*. Engineering Technology. Ph.D.-thesis. 2008, Enschede, The Netherlands: University of Twente.

[14] {QFD Institute}. *The official source for QFD*. 2005 [Last access 12 august 2008]; Available from: http://www.qfdi.org/.

[15] Govers, C.P.M., *What and how about quality function deployment (QFD).* International Journal of Production Economics, 1996. 46-47: p. 575--585.

[16] Finger, S. and J.R. Dixon, *A review of research in mechanical engineering design. Part I: Descriptive, prescriptive, and computer-based models of design processes.* Research in Engineering Design, 1989. 1(1): p. 51--67.

[17] Finger, S. and J.R. Dixon, *A review of research in mechanical engineering design. Part II: Representations, analysis, and design for the life cycle.* Research in Engineering Design, 1989. 1(2): p. 121--137.

[18] Altshuller, G.S., *40 Principles - TRIZ Keys to Technical Innovation*. TRIZ Tools. Vol. 1. 1997, Worcester, MA: Technical Innovation Center.

[19] Manson, S.M., *Simplifying complexity: a review of complexity theory.* Geoforum, 2001. 32(3): p. 405--414.

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

[21] Erden, M.S., et al., *A review of function modeling: Approaches and applications.* AI Edam-Artificial Intelligence for Engineering Design Analysis and Manufacturing, 2008. 22(2): p. 147-169.

[22] Muller, G.J., *CAFCR: A Multi-view Method for Embedded Systems Architecting*. Ph.D.-thesis. 2004, Delft: Delft University of Technology.

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

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

[25] Bonnema, G.M. and P.D. Borches, *Design with Overview – how to survive in complex organisations*, in *18th Annual International Symposium of INCOSE (IS2008)*. 2008, INCOSE: Utrecht, The Netherlands.

[26] León-Rovira, N. and H. Aguayo, *A new Model of the Conceptual Design Process using QFD/FA/TRIZ.* TRIZ-Journal, 1998. 7(July).

[27] Domb, E., *QFD and TIPS/TRIZ.* TRIZ-Journal, 1998. 1998(6).

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