TRIZ FOR SYSTEMS ARCHITECTING

G. Maarten Bonnema
Laboratory of Design, Production and Management,
Department of Engineering Technology,
University of Twente, The Netherlands
e-mail: g.m.bonnema@utwente.nl

Abstract
TRIZ has gained interest over the past decades, as among others expressed by this conference. It is widely used and acknowledged for dealing with technical issues on component level. However, decisions on system level have a much greater impact than those on component level. Thus it is worthwhile to investigate applying TRIZ early in the design process. The article explores the benefits of and possibilities for applying TRIZ in the architecting phase. For this, system architecting is treated in short. An architecting approach presented earlier will be treated and elaborated upon. This approach connects the customer's key drivers with functions to be performed. This approach provides leads for integrating TRIZ in the system architecting phase. These will be discussed in detail as the main subject of the paper. Examples, conclusions and ideas for future work complete the paper.
Keywords: System architecting, Function, Key driver, Method, Coupling matrix, TRIZ

1. Introduction

System architecting will become ever more important as new products have to be created in ever shorter cycles, accompanied by higher functional requirements and increasing multidisciplinarity. Also, the chances of failure have to be reduced from the outset of the project. Therefore more attention in the conceptual phase is required. As French [1985, p.3] states about conceptual design:

> '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'.

However, currently system architecting is not well supported by tools and methods. Most system architects have acquired the knowledge and competences during their carreer as a (system) designer [Muller 2004]. No particular education or specific tools are used by the system designers. Goal of the present research project is to devise a method (and preferably implement that method in a tool) that aids the system designer in creating and evaluating system architectures. In Bonnema [2006] an approach for supporting system architects is presented that uses a *coupling matrix* $\mathcal{C}$, to connect the system functions to the *customer key drivers* described by Muller [2004]. This method, still under development, will be elaborated upon in section 3 as it provides opportunities to connect to TRIZ, after first having looked at what system architecting is in section 2. The strategy is shown in figure 1. This figure shows the general TRIZ approach of generalising the problem and finding a generalised solution. 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 concetrate on in section 4. Section 5 contains examples of application of the method. Section 6 draws conclusions and provides an outlook on future research.

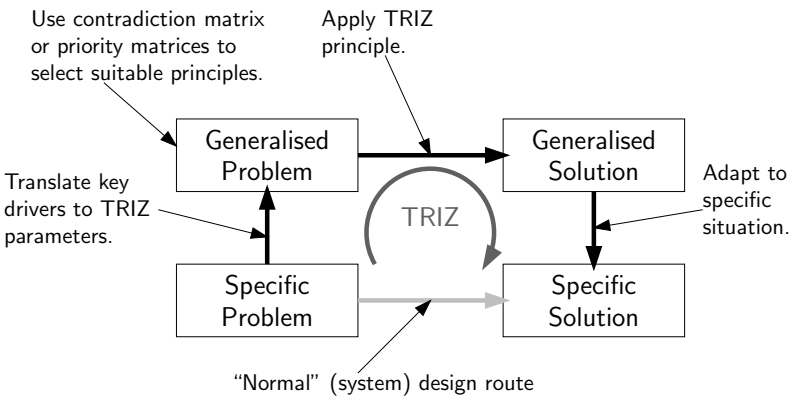

Figure 1: Overview over the approach to be presented. The main contribution is in the first step: generalising the specific problem and selecting suitable TRIZ principles, based on the system architecting information captured with the coupling matrix $\mathcal{C}$.

2. System architecting

In designing complex systems, architecting is an essential step [Maier and Rechtin 2000, Muller 2004]. Complex systems by definition perform many functions. Ensuring proper fit, balance and cooperation between sub-systems in moderately complex systems design can be supervised by one person. For present day and 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 systems architecting is about.

We therefore proposed the following definition [Bonnema 2006]:

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

And thus system architecting is the process of defining a system architecture.

Paradoxically, system's functions *can* be allocated to either its supersystem or the environment. The cooling of a hard disk drive is not performed by the drive itself, but by its supersystem: the computer system it is part of that has a cooling sub-system. This is familiar in TRIZ as "use available resources".

3. FunKey: an architecting method using functions and key drivers.

As mentioned, we have proposed in [Bonnema 2006] a method for system architecting. This method uses a *coupling matrix* $\mathcal{C}$ to connect *fun*ctions to *key* drivers. Therefore, we will call the method *FunKey* from now on. Functions, well known in TRIZ, are tasks to be performed by the system: *expose wafer*, *transport sand*, *create image*.

Key drivers are generalised requirements that express the customers' interest [Muller 2004]. 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.

The FunKey architecting procedure is as follows (see figure 2):

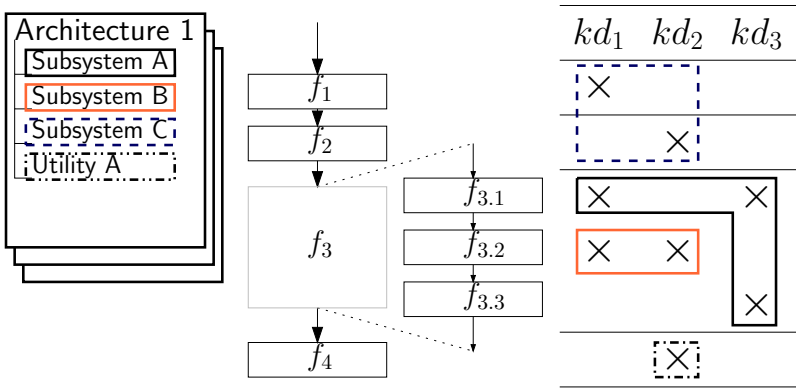

Figure 2: The FunKey architecting method. To the right the coupling matrix $\mathcal{C}$ is shown that connects the functions in the block diagram to the key drivers kd_i. On the left, one architecture is shown. The subsystems are marked in the coupling matrix. On the top level, functions can also be assigned to the user, the environment and the supersystem.

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.

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) [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. This will be used in section 4.

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. 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 the earlier mentioned reference [Bonnema 2006].

4. Connecting TRIZ to FunKey

The information in the FunKey matrix provides information on relations between functions and performance of the entire system. Two strategies will be presented that couple the system information to TRIZ: (1) Using a priority matrix [Ivashkov and Souchkov 2004]; and (2) Using useful and harmful or insufficient and excessive contributions. Each strategy will be elaborated in separate subsections.

For both strategies the key drivers have to be related to the 39 parameters of a technical system, as defined by Altshuller [1997]. These are generalised properties of a system. The key drivers in the FunKey matrix are generalised requirements. Therefore a match between the two seems to be feasible. Problem is that the 39 TRIZ parameters are well established and

Table 1: 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.

Table 2: 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.
iii. Use the contradiction matrix to identify applicable innovative principle(s);
iv. Apply the principle to the function.

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: technology or 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, that will not be treated further apart from the examples provided, FunKey does not differ from normal use of TRIZ.

4.1. Using the priority matrix

Ivashkov and Souchkov [2004] proposed an interesting use of TRIZ in early stages of design, when the amount of available analytical tools is limited. From the well-known contradiction 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}. All scores together, presented in a priority matrix PM^+ indicate the relevance of each of the innovative principles for each parameter. Thus, PM^+ directs the designer to the most successful principle IP_k for improving parameter p_i.

Extending this, we can also 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 priority matrix for worsening features: Instead of counting the number of occurences of IP_k in a *row* of the contradiction matrix, we count the number of occurences in the *column* of parameter p_i.

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 1) is found as in "normal" TRIZ.

4.2. Using useful/harmful or insufficient/excessive contributions

As mentioned in section 3, 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 principles given there can be applied to function f_i.

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^-. The procedure in table 2 has to be modified accordingly.

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.

5. Examples

5.1. 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 4. One of the key drivers of a wafer scanner is *throughput*. In table 3 the main functions of the wafer scanner are shown. Also one column of the FunKey matrix is filled out: the throughput key driver. There are two versions shown: the single case, which was the starting point, and the twin case: the result that can be achieved using TRIZ with FunKey.

Table 3: Connecting functions and key drivers for the wafer scanner case. Only the throughput key driver is shown.

Function	Throughput	
	Single	Twin
Load wafer	×	
Prealign wafer	×	
Wafer to expose chuck	×	
Align wafer	×	
Expose wafer	×	×
Maintain focus		
Position stage	×	×
Unload wafer	×	

One can easily see that most functions influence throughput in the single case. 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.

Let us apply the procedure in section 4.1 to improve throughput. Throughput relates to the TRIZ parameter 39: *productivity*. The priority matrix then suggests to apply innovative principle 10: *prior action* with a score of 20. We can perform all functions but expose wafer and position stage in advance. This is realised in the TwinScan systems by using two simultaneously moving wafer tables [Loopstra et al. 1999]. One performs measurements, the other one exposes a wafer. This results in the FunKey matrix shown in the last column of table 3, which is clearly easier to partition.

5.2. Personal Urban Transporter

Table 4: FunKey for the Personal Urban Transporter (PUT). (conv.: convenience)

Function	$/km	safety	conv.
Maintain posture	–	+	+
Create light			
– on road	–	+	o
– to other traffic	–	+	o
Steer	–	+	–

As second example, part of the Personal Urban Transporter (PUT) introduced in [Bonnema 2006] will be analysed. A PUT is a small, safe and economical vehicle for commuting. Space prohibits a detailed elaboration of the example. Based on an initial analysis of the system, several functions have been assigned to the PUT. In table 4 part of the FunKey table is filled with +es and –es.

We can associate key driver $/km with TRIZ parameter 19: use of energy by moving object, safety with 30: object affected harmful factors, and convenience with 33: ease of operation. For the function maintain posture the contradiction between parameter 30 and 19 is identified, leading to innovative principle 24: mediator. 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 steer, one of the TRIZ principles is 25: self-service. This leads to using the edge of the road to steer the PUT.

6. Conclusions and future work

We have presented a way to connect TRIZ to the architecting method FunKey. This may help the system architect both in finding implementations for his functions (as in example 2), and in simplifying the system (as in example 1). A simpler system, of course, is easier to partition. Also, the principle 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. Both the FunKey method and the linking to TRIZ are currently being tested in industrial cases. Results of these cases will be published in due time.

References

Altshuller, G. S. (1997). *40 Principles - TRIZ Keys to Technical Innovation*, Volume 1 of *TRIZ Tools*. Worcester, MA: Technical Innovation Center.

Bonnema, G. M. (2006). Function and budget based system architecting. In *TMCE 2006*, Ljubljana, Slovenia, pp. 1306–1318.

Chan, L.-K. and M.-L. Wu (2002). Quality function deployment: A literature review. *European Journal of Operational Research 143*(3), 463–497.

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.

French, M. J. (1985). *Conceptual Design for Engineers*. London: Springer-Verlag.

Ivashkov, M. and V. Souchkov (2004). Establishing priority of TRIZ inventive principles in early design. In *Design 2004*, Dubrovnik.

Loopstra, E. R., G. M. Bonnema, H. v. d. Schoot, G. P. Veldhuis, and Y. B. P. Kwan (1999). Lithographic apparatus comprising a positioning device having two object holders. European Patent Office; EP0900412.

Maier, M. W. and E. Rechtin (2000). *The art of systems architecting* (2nd ed.). Boca Raton: CRC Press.

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

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