Transactions of the SDPS:
Journal of Integrated Design and Process Science
XX (2018) XX-XX
DOI 10.3233/jid-2018-0005
http://www.sdpsnet.org

A System Design Reference Model – Based on Design Processes and Designers' Experience

G. Maarten Bonnema (g.m.bonnema@utwente.nl. Tel: (+31)53-4892548)

Department of Design, Production and Management, Faculty of Engineering Technology, University of Twente, Enschede, The Netherlands

Abstract When studying the design process, many models, approaches and methods can be found. As these are all described in different types of diagrams, figures, lists and recipes, comparing them is not an easy task. Also when the results of interviews with or observations from designers have to be ordered, problems may arise. From the study of the design process, and the results of an investigation into the use of models, we will derive a reference model for the system design phase. For this we will also investigate complexity, concreteness and realization.

The reference model can be used to find similarities and dissimilarities between different design processes, and to find possible connections between approaches. We will map several well-known design process models in the reference model.

Keywords: Design Process, Reference Model, System Design, Systems Engineering

1. Introduction

Present state of the art products and systems 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, responsible for a well functioing overall system on the other hand, lacks the detail information needed to supervise the system integrity during the entire process.

In Figure 1 the problem is visualised. While moving from the system (design) level to the component level, the number of issues, items and interfaces to consider increases enormously. Maintaining overview and insight during this development is a great challenge. Because uncertainty is high in the early design process, while the impact of decisions taken is enormous, as shown in Figure 2.

The problem is often approached with classical design methods like those presented in (Alexander, 1966; French, 1985; Kroonenberg and Siers, 1992; Pahl and Beitz, 1996). The reason is that many engineering

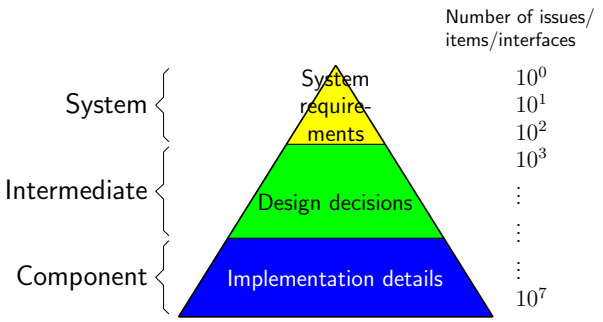


Figure 1. Pyramid showing the increase of issues when moving in the direction of detail design. After (Muller, 2007).

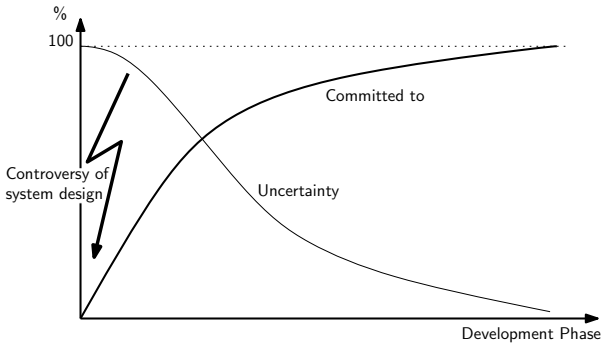


Figure 2. The controversy of system design: decisions with high impact have to be taken early, with high uncertainty. After (Bonnema *et al.*, 2016)

education programs base within a single domain. The design methods taught there are quite able to cope with mono-disciplinary design problems. When confronted with multidisciplinary problems, the professional engineers then return to these methods. In this paper a number of such methods will be presented briefly. In addition, we will show more recently adopted design methods like the spiral development process (Boehm, 1988; Godfrey, 2013), and SIMILAR (Bahill and Gissing, 1997). We will also look at systems thinking, which is often mentioned as an essential skill for system developers (Boardman and Sauser, 2008; Bonnema and Broenink, 2016; Pan, Valerdi and Kang, 2013).

It turns out that the methods and approaches are not entirely equivalent, have different starting points, different levels of abstraction and thus cannot be compared directly. The SDPS society's purpose is "integrating scientific knowledge from diverse disciplines for the purpose of better, more integrated design and improved processes" (Gatchel and Tanik, 2001, p.1). Therefore, we have, in combination with the opinions of expert designers, as reported in (Bonnema and van Houten, 2006), developed a reference model for the system and conceptual design process. The reference model can be used to investigate differences, similarities and possible connections between the methods. It might even help to synthesize new design methods, thus aligning with the SDPS's aim.

A note here on terminology. We focus in this paper on design methods, in particular design methods for multidisciplinary problems. Such methods can be subdivided in prescriptive and descriptive methods, based on whether they guide the designer (prescriptive), or whether they describe what designers do. In particular in the latter case, the description is a *model* of the design process. Therefore, in this paper we sometimes use the term model to refer to a description of the design process. The reference model that is our main deliverable, is not a method nor a prescriptive design process, but a common frame that can be used to compare and connect design process descriptions – both descriptive and prescriptive process models.

We start the discussion with setting the stage (Section 2) and looking at the essence of design, what is abstraction and complexity in designing systems. Then several design process models will be treated in Section 3. These components result in requirements for a system design reference model (Section 4). To arrive at the model, a further investigation into complexity, concreteness and realization is given. The reference model will be presented in Section 5, followed by a mapping of nine design process models in the reference model in Section 6. We end the paper with a discussion (Section 7), conclusions, and directions for further research (Section 8).

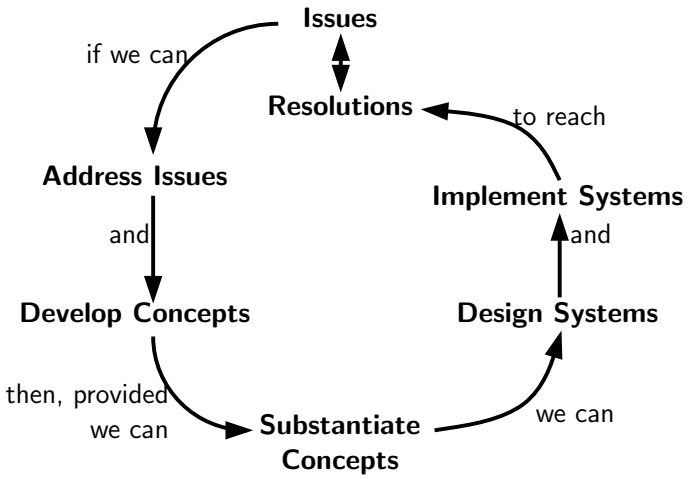


Figure 3. The Systems Design Process according to (Hitchins, 1992, p.xiii).

2. Setting the Stage

2.1. Conditions and Steps in Designing

Before looking at design methods, it is wise to discuss the essence of design. Figure 3 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. The designer has to be enabled to develop concepts. The developed concepts have to be substantiated in order to be of use.

When the conditions mentioned above are met, the design can be detailed, made and implemented. With a proper design, the issue under consideration may be resolved. This of course, has to be verified and tested. One can say that design is cyclic. Every solution provides opportunities for new designs, both in the same type of product and in the surroundings. Think about how the uptake of the smartphone gave rise to an explosion of accessories for smart phones and apps.

A further remark is the increasing number of details. As shown in Figure 1, the number of issues increases with increasing detail. In general, the top level system design is the basis for the next hierarchical level of design. So, matching Hitchins' loop (Figure 3) with Muller's pyramid, urges us to check at each level whether we have resolved the issue, before moving to the next hierarchical level.

The SIMILAR model (State the problem, Investigate alternatives, Model the system, Integrate, Launch the system, Assess performance, and Re-evaluate) for systems engineering (Bahill and Gissing, 1997) also uses a loop to describe the systems design process. An important difference between Hitchins' model and SIMILAR is that the steps in the latter are more concrete.

2.2. Level of Abstraction

As discussed earlier (Bonnema and van Houten, 2004), in (Alexander, 1966, pp. 75–78) three situations are described (see Figure 4). Design in an unselfconscious situation acts directly on the artefact to be designed (see Figure 4a). 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.

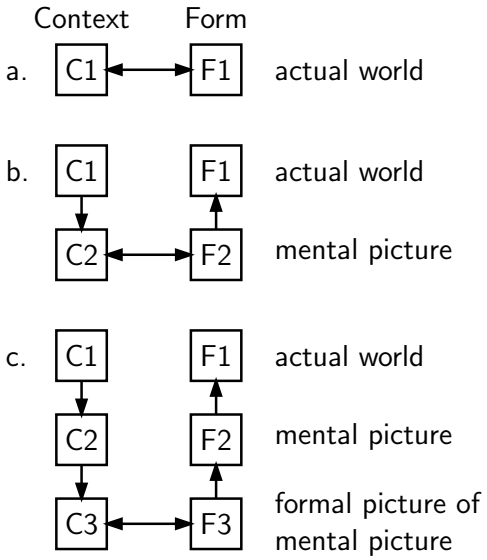


Figure 4. Three design situations according to (Alexander, 1966).

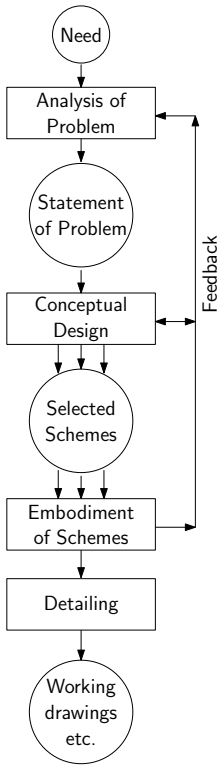


Figure 5. Design Process according to (French, 1985). Rectangles indicate steps, circles indicate results.

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 machines, 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 4b). This conceptual solution is then translated into a form (a product or system[1]). This normally is done by someone else than the designer or the user. The fit can 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.

When the problem is more complex, which is generally the case when designing systems like the ones mentioned in the Introduction, then the designer will not only form a conceptual picture of the context, but also makes a formal model of this conceptual picture (see Figure 4c). Based on this picture, 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 both 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; not many aircraft-designers ever pilot one of their products.

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 grouped in a tree-like structure based

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

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. In present day complex systems however, the requirements space is so large that waiting before the requirements are complete, fixed and organised will lead to a serious lag compared to the competition. Even more so, the requirements change while the system is being developed (Oorschot, 2001).

The view presented in Figure 4 shows the gap many designers face between conceptual idea and realization. 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, also see Figure 2. A pure top-down approach is therefore thought impractical.

2.3. Complexity

As we are discussing the development of complex systems, an elaboration on *complexity* is appropriate. Complexity is widely studied (see for instance (Axelsson, 2002; Casti, 1979; Gell-Mann, 1995; Manson, 2001; Martensson, 1999; Martin, 2013; McDermid, 2000; Nakagawa and Yasui, 2003; Osman, Glass and Hola, 2015; Ottino, 2003; Sch?n and Bennet, 1996; Suh, 2005). It is not our objective to repeat these, but we will use some of those references to explore complexity.

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 on this. (Krumhauer, 1974) only related complexity to the number of objects. (Manson, 2001) distinguishes three types of complexity:

- 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 and defines it as a number C:

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

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 shows that complexity is related to the number of parts, and their different types, and the connections and interfaces, as also reported in (Deshmukh, Talavage and Barash, 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).

In Figure 1, 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 more 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 disentanglement can be created. The types of complexity mainly show up in the intermediate level of the pyramid.

Two types of complexity are identified in (Calvano and John, 2004) where:

- 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, as also treated by (Perrow, 1984). 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.

(Suh, 2005), finally, looks at complexity from an information point of view. He states that complexity relates to uncertainty in meeting the functional requirements. He distinghuishes two main forms of complexity, that are each split in more specific forms:

- time-independent complexity divided into two subtypes:
  - time-independent real complexity: the problem to be solved, or the system to be designed is difficult by nature. An example is the physics involved in magnetic resonance imaging (MRI).
  - time-independent imaginary complexity: “is defined as uncertainty that is not real uncertainty, but arises because of the designer’s lack of knowledge and un- derstanding of a specific design itself.”
- time-dependent complexity, here “future events affect the system in unpredictable ways”. This can be wear in system components, unforeseen system usage, catastrophic events, etc

To sum up, complexity has many faces. The use of the term in normal speech gives rise to confusion, as both difficult and entangled issues are called complex. Design processes need to accommodate complexity, if they are to be applied on system design.

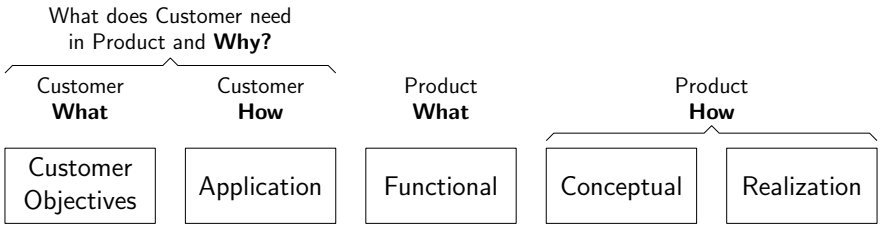


Figure 6. Basic CAFCR Model as presented in (Muller, 2004).

3. Design process models

This section treats several approaches to the early part of the design process, as described in literature. As mentioned earlier, some approach the matter in a descriptive way; telling the designer what he has to do, others merely describe observations they made looking at designers at work. Another scale to use is that between formalised and flexible process models (Murdoch and McDermid, 2000, p.49). That reference seeks a balance between the two, thus supporting innovative design. Although we will not go into ranking the process models, (Pulm and Clarkson, 2005) discusses what constitutes a good process.

This section should not be seen as a complete literature review. The selection of models presented here shows a few rather different views on the matter. Together they do show the extent of the field of research.

3.1. French

Let us first look at a model of the design process proposed in a classic book on conceptual design (French, 1985), see Figure 5. 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.

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.

3.2. CAFCR

In (Muller, 2004) a view on the creation of an architecture of a system is shown: The Customer Objectives, Application, Functional, Conceptual, Realization model, or short CAFCR-model, see Figure 6. The model can be used to describe the views on a system in relation to its context. In Figure 4c the route from context to form runs via mental and formal pictures. The content of these pictures is shown in the CAFCR model in Figure 6: 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 to actual (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.

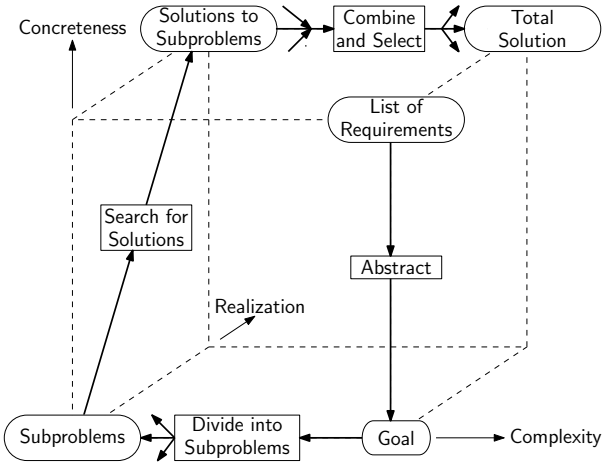


Figure 7. The space of the conceptual design phase (Krumhauer, 1974). Ovals indicate information, rectangles show information processing.

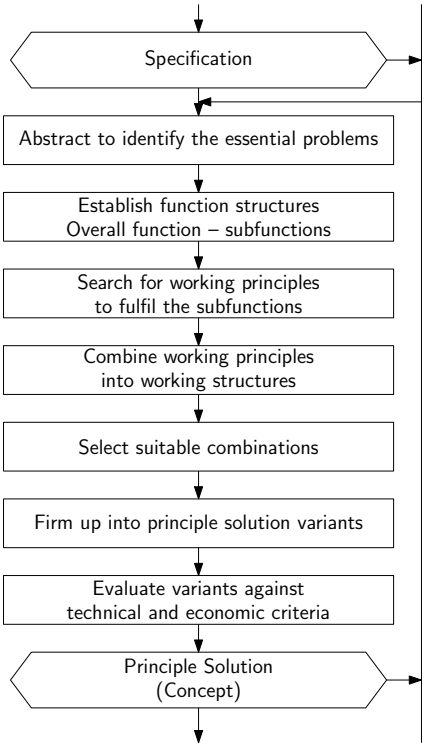


Figure 8. The Conceptual Design Phase according to (Pahl and Beitz, 1996).

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, 2004). An interesting observation is that in Figure 5 the process is described, whereas in Figure 6 the process is left untreated. Instead, the data to be obtained is shown.

3.3. Krumhauer

In (Krumhauer, 1974) the conceptual phase of the design process 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 7. 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 subsolutions 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.

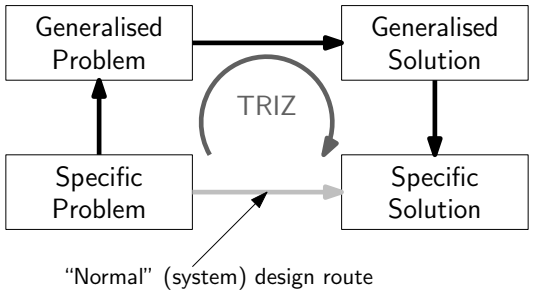


Figure 9. The approach of the Theory of Inventive Problem Solving (TRIZ).

3.4. Pahl & Beitz

In Figure 8 the steps of the conceptual phase according to the well-known classical design process by (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 (reducing complexity). 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 generated from specifications, and one concept is selected.

This process is fairly detailed. Each step shown in Figure 8 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. Later on the focus shifts to finding suitable working principles, and in the end to rating the found concepts.

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 qualitatively as possible and then formulating the problem as neutrally as possible. The problem stated this way can be called the *goal* of the design procedure (also see Figure 7); it contains the essential information, without many distracting details. Please note that the list of requirements that this procedure starts from to has to be quantitative, exact and complete.

3.5. TRIZ

The "Theory of Inventive Problem Solving", known under the Russian acronym TRIZ (Altshuller, 1997,9; Salamatov, 1999) was invented by the Russian engineer Genrich Altshuller. It states, among others, that technical problems can be resolved using a limited set of patterns. These patterns have been derived from investigation of thousands of (Russian) patents. The general approach is shown in Figure 9. The problem at hand is translated into a generalised problem that is classified either by reformulating it into a contradiction, rating it on the scales of evolution or modelling it in the 9-window diagram, among others. For each approach 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.

Although the 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. In general, TRIZ is used for problems that are not complex (as in time-independent imaginary complexity). It is a powerful technique for dealing with time-independent real complex problems.

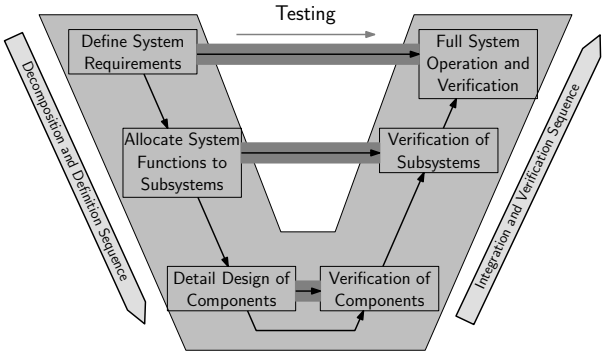


Figure 10. The "Vee"-process model for systems engineering (Blanchard and Fabrycky, 2011).

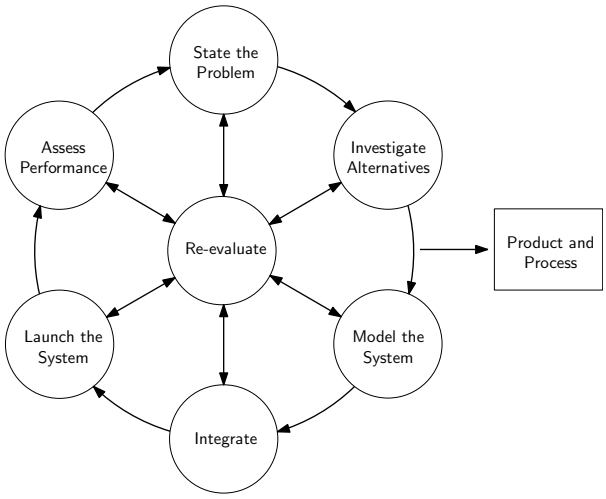


Figure 11. The SIMILAR process for systems engineering (Bahill and Gissing, 1997).

3.6. "Vee" model

The well-known "Vee"-model, see Figure 10 (Blanchard and Fabrycky, 2011), shows how the development of systems connects to integration and validation of the design. On system level, the system 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, that is going downward in the Vee, the coupling between creation and qualification is made by compiling the test specifications. Striking omission in the classic Vee-model is the user (or customer) wish: what is it that he really needs, and how can we prove in the end that the development has met that wish? Therefore we extended the Vee-model in (Bonnema *et al.*, 2016).

3.7. SIMILAR

In Section 2.1, we already mentioned the SIMILAR model for systems engineering (Bahill and Gissing, 1997). The authors gave a mere textual description of their approach, but later visualisations were made, that show the SIMILAR process in a circular loop, with re-evaluate (originally reengineer) as hub, see Figure 11. All actions performed, lead to constant re-evaluation of the goal attainment. As for the Investigate Alternatives step, this is detailed as Establish Goals and Determine Strategies.

3.8. Systems Thinking

Systems thinking is a broad field with different views, opinions and approaches. (Cabrera, 2006) has presented an extensive overview of the field. Others have looked at it (Boardman and Sauser, 2008; Pan *et al.*, 2013; Sillitto, 2012) to aid developers of complex systems. We developed using (Richmond, 1993) a set of thinking tracks that concretize the thinking. These thinking tracks, as do all systems thinking approaches, assist in exploring the entire design space. In (Bonnema and Broenink, 2016) we showed that the tracks cover all the systems thinking aspects, brought forward by prominent systems thinkers. The tracks are listed in Table 1.

3.9. Spiral Product Development

The spiral development process, originally introduced by Barry Boehm for software engineering[2] (Boehm, 1988), and widely adopted later (Godfrey, 2013), sets itself apart from the traditional waterfall process. Instead, a series of fast loops of development, show through a series of prototypes quickly what the process is working towards, and involve stakeholders. The loops of the spiral development process each go through:

(1). Determine objectives, alternatives, constraints;

(2). Evaluate alternatives, identify and resolve risks;

(3). Develop, verify next-level product; and

(4). Plan next phase.

One of its main accomplishments is reduction of risks. Today, the spiral process is adopted widely, also outside of the purely software domain.

4. Reference Model Development

In this Section we will derive a reference model for conceptual and system design (so broader than pure software engineering) that can be used to map process driven approaches and data driven approaches. The basis is formed by the processes treated in Section 3. In addition, we have interviewed designers (reported earlier in (Bonnema and van Houten, 2006)), where we discussed conceptual and systems design. The (product) models used by these designers, provide information on what aspects need to be covered by such a system design reference model.

4.1. Krumhauer Revisited

The model in (Krumhauer, 1974), see Figure 7, 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 8). In (Bonnema and van Houten, 2006) it was concluded that the interviewed conceptual designers did recognise the space and the general route by Krumhauer.

Krumhauer prescribes a start from a list of requirements, then to abstract to come to the goal of the project. The interviewed designers expressed 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

[2]Note that the present paper is about the broader field of *systems engineering*.

Table 1. The 12 systems thinking tracks (Bonnema and Broenink, 2016; Bonnema *et al.*, 2016)

Dynamic Thinking	Decomposition-Composition Thinking
Feedback Thinking	Hierarchical Thinking
Specific-Generic Thinking	Organizational Thinking
Operational Thinking	Life-Cycle Thinking
Scales Thinking	Safety Thinking
Scientific Thinking	Risk Thinking

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.

4.2. Hitchins, Alexander, French, and CAFCR Revisited

The models by Hitchins (1992), Alexander (1966) and French (1985) and the CAFCR model (Muller, 2004) 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.

(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.

4.3. Revisiting systems design and engineering approaches

SIMILAR is a loop, like Hitchins' model. Here, as in spiral development, the emphasis is on development in short cycles. While working on delivering a system, the design is constantly being (re-)evaluated and adapted where needed. Both SIMILAR and the spiral model show that feedback into the development process should not wait until the system is ready. Already early versions and prototypes can be used for learning and making adjustments, as we can also find in *agile* approaches to software (and systems) development.

Systems thinking, on the other hand, is not so much an approach to development. It describes and creates the right mindset in the developers. Nevertheless, it is our aim to also show this mindset in the reference model. That means that various views (like the ones in Table 1) should be mapped.

4.4. Requirements for the Reference Model

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

- Allow for iteration;
- Allow for abstraction;
- Allow for different routes through the design space;
- Allow for hierarchical designs (and thus allow for recurrence);
- Allow for (systems) thinking directions.

4.5. The Reference Model Space

A space like introduced by Krumhauer will be used 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 3.3). 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

Complexity has many faces, as treated in Section 2.3. The overarching aspect of complexity is that of entanglement: how much interaction is there between the parts/aspects/issues in the system design space. While complete uncoupling is impossible (and often not wanted despite Suh's first axiom (Suh, 1998)), a general approach to dealing with complex problems is disentanglement. We therefore propose a scale ranging from simplex to composite. This way, we avoid confusion about the broad meaning of complexity.

4.5.2. Concreteness

According to (Krumhauer, 1974) and the way we have used the concreteness scale (Bonnema and van Houten, 2006), 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.

(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. The axis will therefore range from concrete to abstract.

4.5.3. Realization

In (Bonnema and van Houten, 2006) we used the realization scale (from idea to product) to track the progress in a design project. The realization scale can 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 "Vee"-model shows how realization comes into system design (Figure 10). In our reference model, we will not use a realization scale, but a scale that shows the progress from vague idea (mental) to actual product (actual), in line with (Alexander, 1966).

5. The Reference Model

Based on the above, we will use three perpendicular bipolar scales:

- Simplex ↔ Composite;

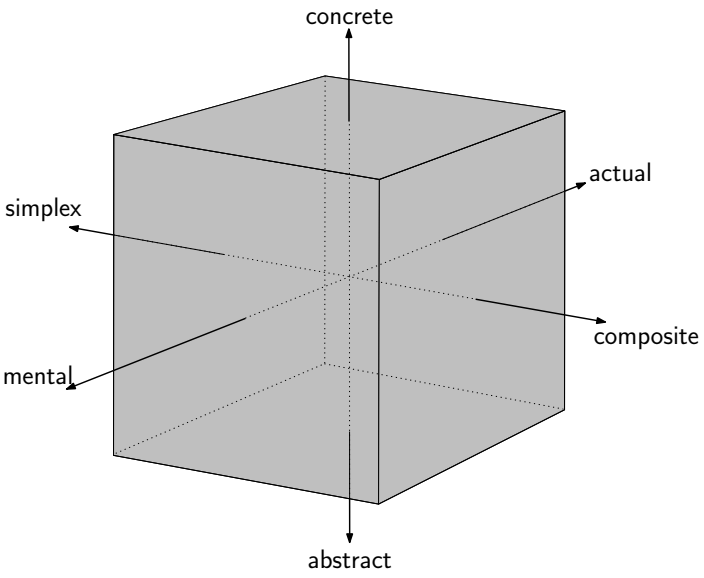


Figure 12. The conceptual and system design reference model.

- Abstract ↔ Concrete;
- Mental ↔ Actual.

Figure 12 shows a visualisation of the reference model, where the space is visualised as a cube, chosen for correspondence with the Krumhauer model.

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. Existing tools and methods can be put in this space so that a wise combination of tools can be made within a chosen method for the problem at hand.

Also both data-based models, like the CAFCR model (Figure 6) and the six-box TRIZ model (Nakagawa, 2006), and action-based models, like the models by Pahl & Beitz (Figure 8), can be visualised, as we will see next.

6. Design Processes and the Reference Model

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

(1). Krumhauer, Figure 13(a);

(2). Pahl & Beitz, Figure 13(b);

(3). TRIZ, Figure 13(c);

(4). Christopher Alexander, Figure 13(d);

(5). CAFCR, Figure 13(e);

(6). Vee-model, Figure 13(f);

(7). SIMILAR, Figure 13(g);

(8). Spiral development, Figure 13(h); and

(9). Systems Thinking tracks, Figure 14, and System Design Tools, Figure 15.

In the following we will only visualise the routes. The distinct steps and intermediate results are not shown.

6.1. Krumhauer

The route by Krumhauer (Figure 7) through the design space can be directly copied. It is striking that concretization and actualisation occur simultaneously.

6.2. Pahl & Beitz

The model by Pahl & Beitz (Figure 8) 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 means 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 movement along the simplex–composite axis. Figure 13(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 13(b) represents the evaluation.

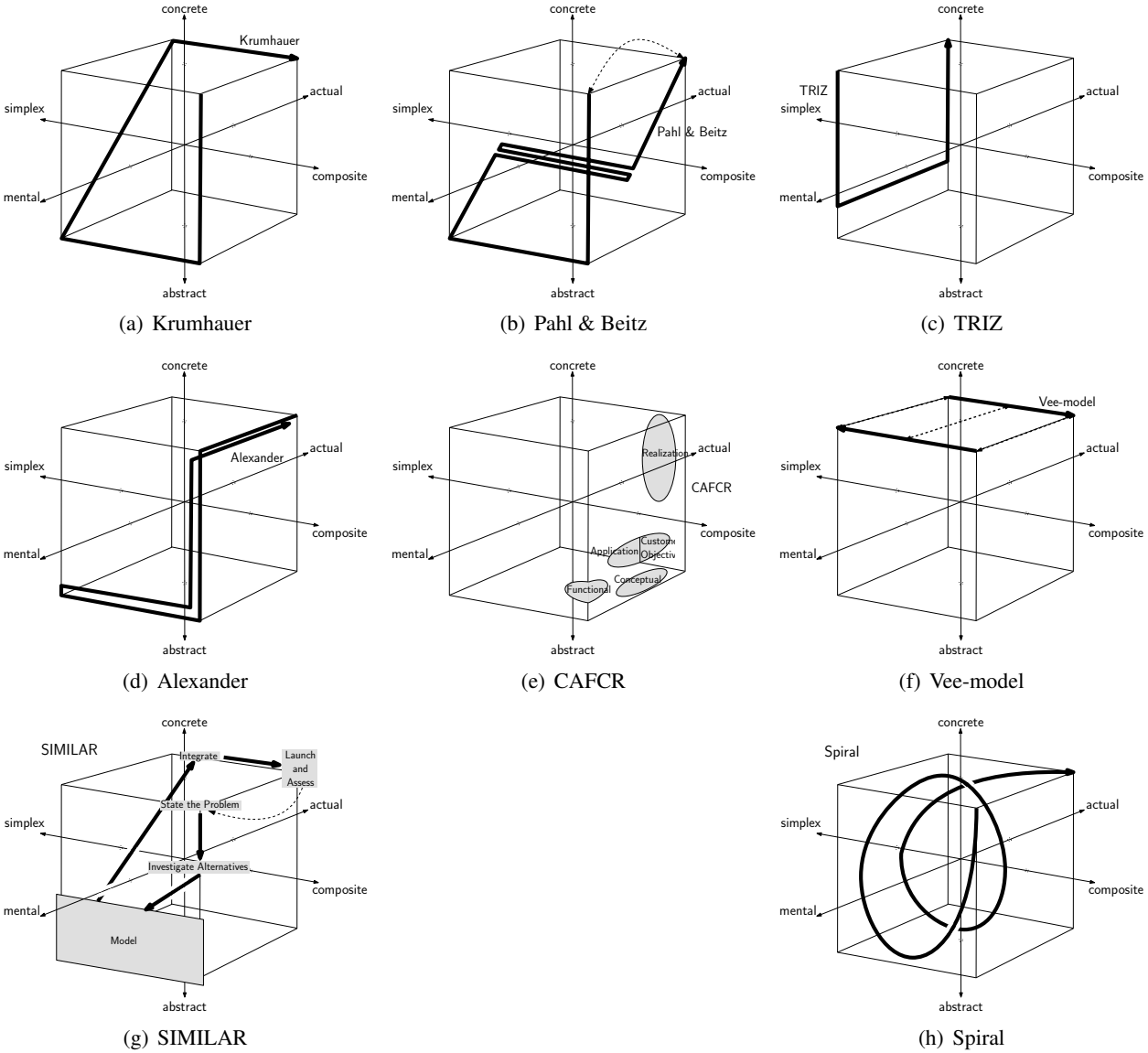
6.3. TRIZ

TRIZ is seen as a conceptual design method. TRIZ by itself, though, cannot deal with composite problems. Characteristic for TRIZ is a detour, see Figure 9. 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 13(c). This detour is present in both the four-box (Figure 9) and the six-box representation (Nakagawa, 2006). Both are visualised in the reference model by the same route in Figure 13(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.

6.4. 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 4). We can model the situation in the complex, self-conscious situation (Figure 4(c)) by the scheme in Figure 13(d). The move along the bottom left edge and back is not taken from the model in Figure 4, but from the text in (Alexander, 1966): The solution for the described problem is to



(a) Krumhauer (b) Pahl & Beitz (c) TRIZ

(d) Alexander (e) CAFCR (f) Vee-model

(g) SIMILAR (h) Spiral

Figure 13. Different design process models in the conceptual and system design reference model.

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.

6.5. CAFCR

The Customer Objectives, Application, Functional, Conceptual and Realization (CAFCR) model presented in Figure 6 does not describe a process with 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 13(e). Note that they are all placed in or near the composite face of the model.

6.6. Vee-model

The Vee-model shown in Figure 10 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.

6.7. SIMILAR

Figure 13(g) shows how the SIMILAR model fits in the reference model space. The model part of SIMILAR can extend along different axes, shown by the grey area. With techniques like hardware in the loop and software in the loop, the modelling can also extend along the mental-actual axis. Not shown is re-evaluate from the SIMILAR model. In fact, that is – as symbolized by the hub in Figure 11 – a constant activity, which all other activities should report to.

Note that in the SIMILAR model there is a direct link from Model the system, to Integrate the system. There is no intermediate step on realization (design and engineering) of parts or subsystems that can be tested and verified prior to integration.

6.8. Spiral Development

The spiral in Figure 13(h) shows only two loops, but of course, multiple loops that work towards a concrete and composite solution can occur. Also, the exact position of the loops can be shifted, depending on the current development situation. In the figure, we show a composite start, but cases where the original seed development is not composite nor complex exist. Then, complexity increases with the sequence of development loops. Multiple concretizations are characteristic for spiral development.

6.9. Thinking Tracks

There is a difference in the process models treated above, and systems thinking. As said in Section 3.8, systems thinking is more a state of mind during systems development. In Figure 14 we chose to show four thinking tracks: Operational, Decomposition-Composition, Life-Cycle and Hierarchical Thinking.

Not all thinking tracks can be precisely positioned in the model. While one may object that the reference model is not complete, we like to pose that as the thinking tracks show states of mind, and aspects that play a

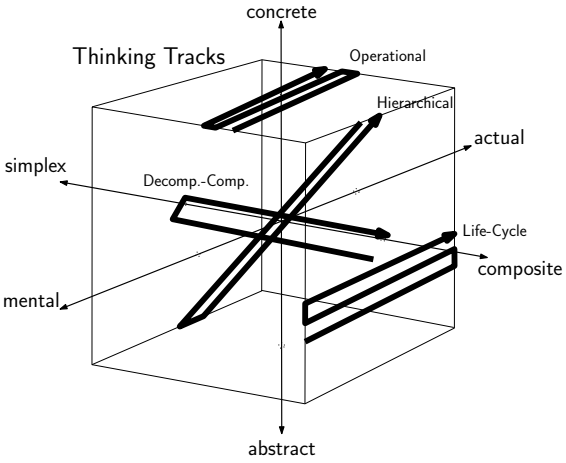


Figure 14. Four Thinking Tracks in the Reference Model space

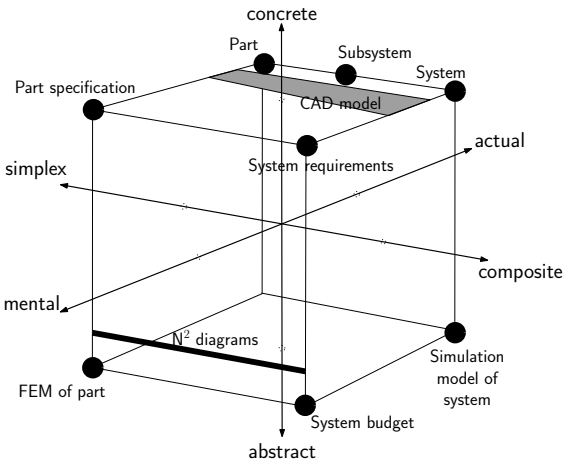


Figure 15. The system design reference model with a few system design tools, parts, subsystems and the position of the final system.

role in systems design, these do not have to be modelled in the design process oriented reference model. The thinking tracks shown, represent directions for thinking. Actual start and end positions may vary depending on where in the design process the tracks are employed.

6.10. Systems Design and Engineering Tools

The systems designer's toolbox contains a large number of tools to use in systems design and engineering (Bonnema *et al.*, 2016; Kapurch and Rainwater, 2007; INCOSE SEH Working Group, 2015). The idea is that the reference model can be used to match process with tools and thinking. In Figure 15 we show some of those tools (Budget, Finite Element Model (FEM), Requirements) and the realized part, subsystem and system.

7. Discussion

There is a large variety of design process models that relate to systems design. The objective of this paper is not to evaluate these models, methods or approaches, but we tried to make a common reference. The idea was that by using such a common reference, the models, methods and approaches can be more easily compared. Also, depending on the starting point of the development process, one approach may be more suitable than another.

The work by (Krumhauer, 1974) inspired us to use a three-dimensional space for such a reference model. Some modification of the axes was necessary to make the model better fit to system design. Note that the three axes we defined (simplex $\leftrightarrow$ composite, mental $\leftrightarrow$ actual, and abstract $\leftrightarrow$ concrete) are used as qualitative measures. Also, we showed the space from the same perspective in all figures, see Figure 12). It can be interesting to experiment further with quantification of the scales, and rotating the space to highlight certain aspects of design processes.

Looking at the mapped design processes in Figure 13, we see that with the exception of TRIZ, the end point is in all cases concrete, actual and composite. Yet, we see quite some variation in the starting point. This means that if a quick scan is done on the starting point of an actual design problem, the reference model can aid in selecting an appropriate approach.

Looking at Figure 15, and comparing to Figure 13), we see that for instance the N^2 diagram, can be a useful tool in the processes by Krumhauer, Pahl& Beitz, Alexander and in SIMILAR. Hierarchical Thinking (Figure 14) can be useful in those processes as well, to connect different phases of the process. Similar reasoning can be applied to other thinking tracks and tools, and possibly the software tools that base upon those.

Reviewing the requirements for the model (Section 4.4), we can easily state that abstraction, different routes and thinking directions are accommodated. Iteration is possible, but the models mapped in Figure 13 do not all model that. So while the reference model allows for it, it depends on the actual approach whether it will occur. The same holds for recurrence.

8. Conclusions

We can conclude that the reference model provides a tool for visualising different approaches to the system 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.

We have shown the routes and data described by different design process models in the reference model. 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. SIMILAR and the spiral model show similarities, and systems thinking can aid in understanding the design space.

Existing system development tools can be employed depending on the place in the process. The fit between tool and process can be investigated with the reference model. 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. Such a tool 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. 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.

9. Recommendations

The reference model shown in the present paper can be used for comparing other design approaches and tools than the ones shown in Figures 13–15. This may lead to possible connections between, or assemblies of different tools and methods in a more integrated approach to design.

Research in specific parts of the design process can use the model to visualise the way of working of designers. This can be very helpful in protocol studies.

Further, more tools in the systems design and engineering domain should be positioned in the reference model space, to see how tools interact, overlap or are disconnected. This will be one line of future research. Another line of research is to employ the reference model to find points for connecting software tools (Endig and Jesko, 2001).

References

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

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.

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.

Bahill, A. T. and B. Gissing: 1997; *No Matter What the Application, It Is Still Systems Engineering*; **INCOSE International Symposium**; **vol. 7** (1): pp. 348–355.
http://dx.doi.org/10.1002/j.2334-5837.1997.tb02192.x

Blanchard, B. S. and W. J. Fabrycky: 2011; Systems Engineering and Analysis; Prentice Hall, Upper Saddle River, New Jersey; 5th edn.
http://www.pearsoned.co.uk/bookshop/detail.asp?item=100000000374477

Boardman, J. and B. Sauser: 2008; Systems Thinking - Coping with 21st Century Problems; Manufacturing & Industrial Engineering; CRC Press, Boca Raton.
http://www.crcnetbase.com/isbn/9781420054927

Boehm, B. W.: 1988; *A Spiral Model of Software Development and Enhancement*; **Computer**; **vol. 21** (5): pp. 61–72.
https://doi.org/10.1109/2.59

Bonnema, G. M. and J. F. Broenink: 2016; *Thinking Tracks for Multidisciplinary System Design*; **Systems**; **vol. 2016** (4(4)).
http://www.mdpi.com/2079-8954/4/4/36

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.
http://home.ctw.utwente.nl/bonnemagm/pubs/ConceptueelOntwerpenCIRP2003.pdf

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., K. T. Veenvliet and J. F. Broenink: 2016; Systems Design and Engineering: Facilitating Multidisciplinary Development Projects; CRC Press, Boca Raton, FL.
http://tinyurl.com/SDE-book

Cabrera, D. A.: 2006; Systems Thinking; PhD-thesis; Cornell University.
https://ecommons.cornell.edu/handle/1813/2860

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.

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

Endig, M. and D. Jesko: 2001; *Engineering Processes - on an Approach to Realize a Dynamic Process Control*; **Journal of Integrated Design & Process Science**; **vol. 5**: pp. 65–82.
https://content.iospress.com/download/journal-of-integrated-design-and-process-science/jid5-2-06?id=journal-of-integrated-design-and-process-science%2Fjid5-2-06

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

Gatchel, S. G. and M. M. Tanik: 2001; *Process Science and Philosophy*; **Journal of Integrated Design & Process Science**; **vol. 5**: pp. 1–21.
https://content.iospress.com/download/journal-of-integrated-design-and-process-science/jid5-4-01?id=journal-of-integrated-design-and-process-science%2Fjid5-4-01

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

Godfrey, P.: 2013; *Architecting Complex Systems in New Domains and Problems: Making Sense of Complexity and Managing Its Unintended Consequences*; in Complex Systems Design & Management, edited by M. Aiguier, Y. Caseau, D. Krob and A. Rauzy; pp. 41–51; Springer Berlin Heidelberg.
http://dx.doi.org/10.1007/978-3-642-34404-6_3

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/

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

Kapurch, S. J. and N. E. Rainwater: 2007; Nasa Systems Engineering Handbook Sp/-2007-6105 Rev1; NASA.
http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20080008301_2008008500.pdf

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

Krumhauer, P.: 1974; Rechnerunterst?tzung F?r Die Konzeptphase Der Konstruktion: Ein Beitrag Zur Entwicklung Eines Programmsystems F?r Die L?sungsfindung Konstruktiver Teilaufgaben; PhD-thesis; Technischen Universitaet Berlin.

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

Martin, J. N.: 2013; *Managing Complexity and Change Using Architecture Frameworks & Modeling*; Course Material.

INCOSE SEH Working Group: 2015; Systems Engineering Handbook; INCOSE; 4th edn.

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

Muller, G. J.: 2004; Cafcr: A Multi-View Method for Embedded Systems Architecting; PhD-thesis; Delft University of Technology.
http://www.gaudisite.nl/Thesis.htmlhttp://www.gaudisite.nl/ThesisBook.pdf

Muller, G. J.: 2007; *Design Objectives and Design Understandability*; Last accessed on 13 May 2011.
http://www.gaudisite.nl/DesignUnderstandabilitySlides.pdf

Murdoch, J. and J. A. McDermid: 2000; *Modelling Engineering Design Processes with Role Activity Diagrams*; **Journal of Integrated Design & Process Science**; **vol. 4**: pp. 45–65.
https://content.iospress.com/download/journal-of-integrated-design-and-process-science/jid4-2-04?id=journal-of-integrated-design-and-process-science%2Fjid4-2-04

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

Oorschot, v. K. K.: 2001; Analyzing Npd Projects from an Operational Control Perspective; Ph.D. thesis; Technische Universiteit Eindhoven.
http://repository.tue.nl/549472

Osman, M., B. D. Glass and Z. Hola: 2015; *Approaches to Learning to Control Dynamic Uncertainty*; **Systems**; **vol. 3** (4): p. 211.
http://www.mdpi.com/2079-8954/3/4/211

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

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

Pan, X., R. Valerdi and R. Kang: 2013; *Systems Thinking: A Comparison between Chinese and Western Approaches*; in Conference on Systems Engineering Research (CSER13); pp. 1027–1035; Procedia Computer Science, Atlanta.

Perrow, C.: 1984; Normal Accidents: Living with High-Risk Technologies; Basic Books, New York.
https://en.wikipedia.org/wiki/Normal_Accidents

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

Pulm, U. and J. Clarkson: 2005; *The Meaning and Coherence of Process, Organisation and Product in Engineering Design*; **Journal of Integrated Design & Process Science**; **vol. 9**: pp. 55–66.
https://content.iospress.com/download/journal-of-integrated-design-and-process-science/jid9-1-05?id=journal-of-integrated-design-and-process-science%2Fjid9-1-05

Richmond, B.: 1993; *Systems Thinking: Critical Thinking Skills for the 1990s and Beyond*; **System Dynamics Review**; **vol. 9** (2): pp. 113–133.
http://dx.doi.org/10.1002/sdr.4260090203

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

Salamatov, Y. P.: 1999; Triz: The Right Solution at the Right Time: A Guide to Innovative Problem Solving; Insytec, Hattem, The Netherlands.

Sch?n, 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

Sillitto, H.: 2012; *Integrating Systems Science, Systems Thinking, and Systems Engineering: Understanding the Differences and Exploiting the Synergies*; in International Symposium of the International Council

of Systems Engineering (INCOSE 2012); INCOSE, Rome.
https://www.incose.org/ipub/12/36.PDF

Suh, N. P.: 1998; *Axiomatic Design Theory for Systems*; **Research in Engineering Design**; **vol. 10** (4): pp. 189–209.
http://dx.doi.org/10.1007/s001639870001

Suh, N. P.: 2005; *Complexity in Engineering*; **Annals of CIRP**; **vol. 2** (2005): pp. 581–598.