

# Module System Architect Toolkit



Gerrit Muller

University of South-Eastern Norway-NISE

Hasbergsvei 36 P.O. Box 235, NO-3603 Kongsberg Norway

[gaudisite@gmail.com](mailto:gaudisite@gmail.com)

## Abstract

This module addresses tools and techniques available to the System Architect. It explains the basic CAFCR method and addresses story telling as method.

### **Distribution**

This article or presentation is written as part of the Gaudí project. The Gaudí project philosophy is to improve by obtaining frequent feedback. Frequent feedback is pursued by an open creation process. This document is published as intermediate or nearly mature version to get feedback. Further distribution is allowed as long as the document remains complete and unchanged.

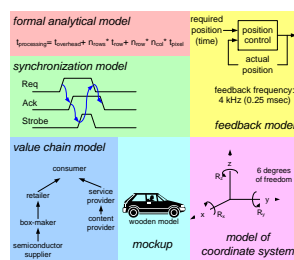
All Gaudí documents are available at:  
<http://www.gaudisite.nl/>

# Contents

<b>1</b>	<b>Basic Working Methods of a System Architect</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Viewpoint hopping . . . . .	3
1.3	Decomposition and integration . . . . .	7
1.4	Quantification . . . . .	8
1.5	Coping with uncertainty . . . . .	10
1.6	Modelling . . . . .	12
1.7	WWHWWW questions . . . . .	14
1.8	Decision Making Approach in Specification and Design . . . . .	15
1.9	Acknowledgements . . . . .	17
<b>2</b>	<b>Short introduction to basic “CAFCR” model</b>	<b>18</b>
2.1	Introduction . . . . .	18
2.2	The CAFCR model . . . . .	19
2.3	Who is the customer? . . . . .	19
2.4	Life Cycle view . . . . .	20
<b>3</b>	<b>Story How To</b>	<b>23</b>
3.1	Introduction . . . . .	23
3.2	How to Create a Story? . . . . .	24
3.3	How to Use a Story? . . . . .	25
3.4	Criteria . . . . .	25
3.5	Example Story . . . . .	27
3.6	Acknowledgements . . . . .	28

# Chapter 1

## Basic Working Methods of a System Architect



### 1.1 Introduction

The basic working methods of the architects are covered by a limited set of very generic patterns:

- Viewpoint hopping, looking at the problem and (potential) solutions from many points of view, see section 1.2.
- Decomposition, breaking up a large problem in smaller problems, introducing interfaces and the need for integration, see section 1.3.
- Quantification, building up understanding by quantification, from order of magnitude numbers to specifications with acceptable confidence level, see section 1.4.
- Decision making when lots of data is missing, see section 1.5.
- Modelling, as means of communication, documentation, analysis, simulation, decision making and verification, see section 1.6.

- Asking Why, What, How, Who, When, Where questions, see section 1.7.
- Problem solving approach, see section 1.8.

Besides these methods the architect needs lots of “soft” skills, to be effective with the large amount of different people involved in creating the system. See [6], [2] and [3] for additional descriptions of the work and skills of the architect.

## 1.2 Viewpoint hopping

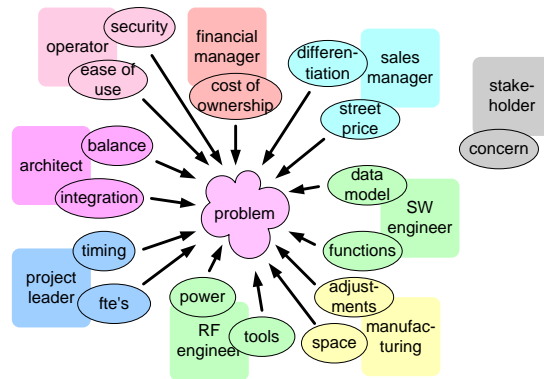


Figure 1.1: Small subset of viewpoints

The architect is looking towards problems and (potential) solutions from many different viewpoints. A small subset of viewpoints is visualized in figure 1.1, where the viewpoints are shown as stakeholders with their concerns.

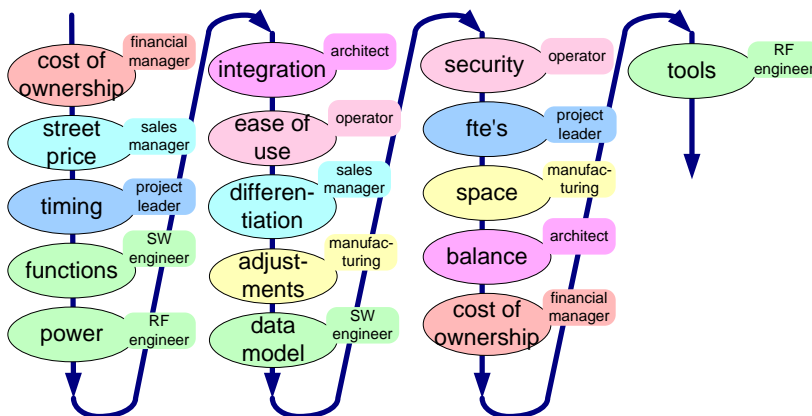


Figure 1.2: Viewpoint Hopping

The architect is interested in an overall view on the problem, where all these viewpoints are present simultaneously. The limitations of the human brains force the architect to create an overall view by quickly alternating the individual viewpoints. The order in which the viewpoints are alternated is chaotic: problems or opportunities in one viewpoint trigger the switch to a related viewpoint. Figure 1.2 shows a very short example of viewpoint hopping. This example sequence can

take anywhere from minutes to weeks. In a complete product creation project the architect makes thousands<sup>1</sup> of these viewpoint changes.

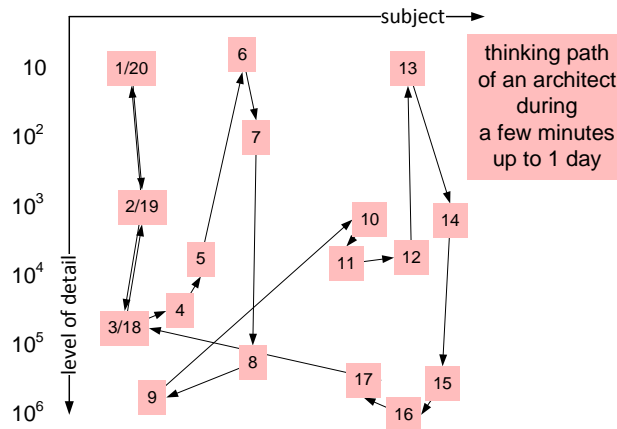


Figure 1.3: The seemingly random exploration path

Viewpoint hopping is happening quite fast in the head of the architect. Besides changing the viewpoint the architect is also zooming in and out with respect to the level of detail. The dynamic range of the details taken into account is many orders of magnitude. Exploring different subjects and different levels of detail together can be viewed as an exploration path. The exploration path followed by the architect (in the architect's head) appears to be quite random. Figure 1.3 shows an example of an exploration path happening inside the architects head.

The plane used to show the exploration path has one axis with *subjects*, which can be stakeholders, concerns, functions, qualities, design aspects, et cetera, while the other axis is *the level of detail*. A very coarse (low level of detail) is for example the customer key driver level (for instance cost per placement is 0.1 milli-cent/placement). Examples at the very detailed level are lines of code, cycle accurate simulation data, or bolt type, material and size.

Both axis span a tremendous dynamic range, creating a huge space for exploration. Systematic scanning of this space is way too slow. An architect is using two techniques to scan this space, that are quite difficult to combine: open perceptive scanning and scanning while structuring and judging. The open perceptive mode is needed to build understanding and insight. Early structuring and judging is dangerous because it might become a self-fulfilling prophecy. The structuring and judging is required to reach a result in a limited amount of time and effort. See figure 1.4 for these 2 modes of scanning.

The scanning approach taken by the architect can be compared with *simulated*

<sup>1</sup>Based on observations of other architects and own experience.

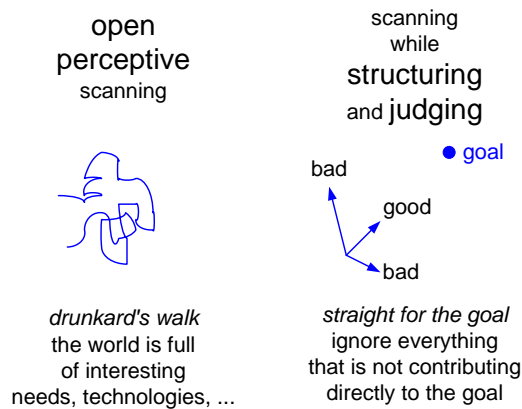


Figure 1.4: Two modes of scanning by an architect

*annealing methods* for optimization[5]. An interesting quote from this book, comparing optimization methods:

Although the analogy is not perfect, there is a sense in which all of the minimization algorithms thus far in this chapter correspond to rapid cooling or quenching. In all cases, we have gone greedily for the quick, nearby solution: From the starting point, go immediately downhill as far as you can go. This, as often remarked above, leads to a local, but not necessarily a global, minimum. Nature's own minimization algorithm is based on a quite different procedure...

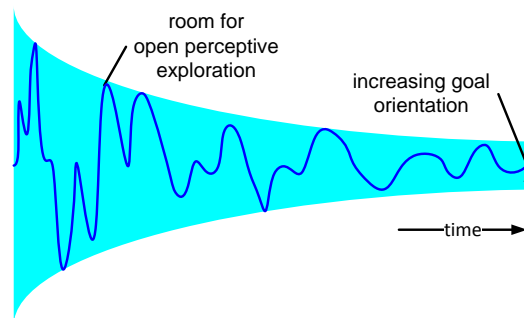


Figure 1.5: Combined open perceptive scanning and goal-oriented scanning

See also figure 1.5 for the combined scanning path. The perceptive mode is used more early in the project, while at the end of the project the goal oriented mode is dominant.

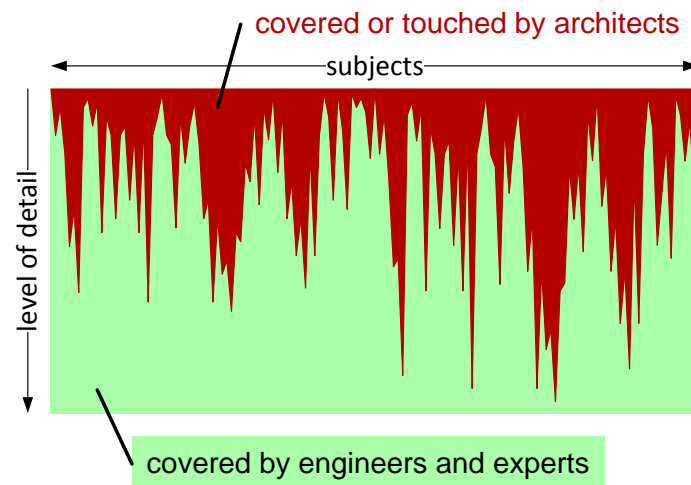


Figure 1.6: The final coverage of the problem and solution space by architect and engineers

The coverage of the problem and solution space is visualized in figure 1.6. Note that the area covered or touched by the architect(s) is not exclusively covered, engineers will also cover or touch that area partially. The architect needs experience to learn when to dig deeper and when to move on to next subjects. Balancing depth and breadth is still largely an art.



## 1.4 Quantification

The architect is continuously trying to improve his understanding of problem and solution. This understanding is based on many different interacting insights, such as functionality, behavior, relationships et cetera. An important factor in understanding is the **quantification**. Quantification helps to get grip on the many vague aspects of problem and solution. Many aspects can be quantified, much more than most designers are willing to quantify.

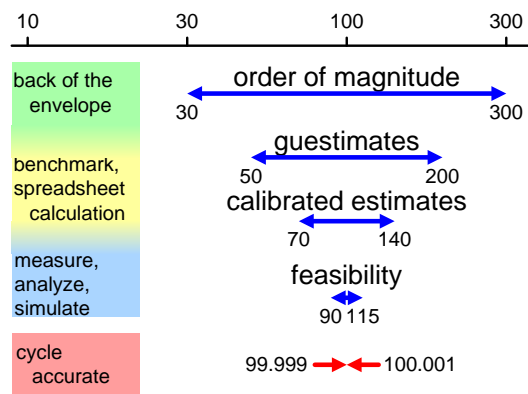


Figure 1.8: Successive quantification refined

The precision of the quantification increases during the project. Figure 1.8 shows the stepwise refinement of the quantification. In first instance it is important to get a feeling for the problem by quantifying orders of magnitude. For example:

- How large is the targeted customer population?
- What is the amount of money they are willing and able to spend?
- How many pictures/movies do they want to store?
- How much storage and bandwidth is needed?

The order of magnitude numbers can be refined by making back of the envelop calculations, making simple models and making assumptions and estimates. From this work it becomes clear where the major uncertainties are and what measurements or other data acquisitions will help to refine the numbers further.

At the bottom of figure 1.8 the other extreme of the spectrum of quantification is shown, in this example cycle accurate simulation of video frame processing results in very accurate numbers. It is a challenge for an architect to bridge these worlds.

Figure 1.9 shows an example how the quantification evolves in time. The dotted red line represents the required performance as defined in the specification. The

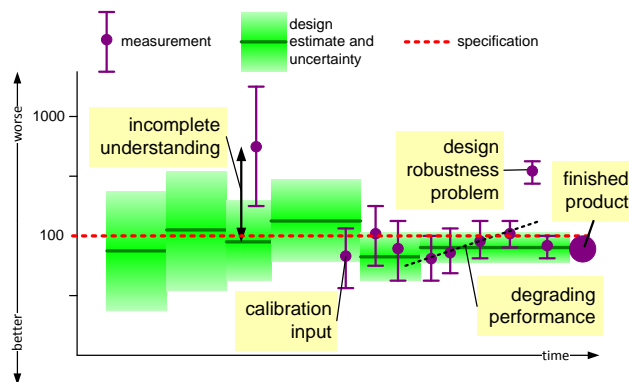


Figure 1.9: Example of the evolution of quantification in time

shaded area indicates the “paper” value, with its accuracy. The measurements are shown as dots with a range bar. A large difference between paper value and measurement is a clear indication of missing understanding. Later during the implementation continuous measurements monitor the expected outcome, in this example a clear degradation is visible. Large jumps in the measurements are an indication of a design which is not robust (small implementation changes cause large performance deviations).

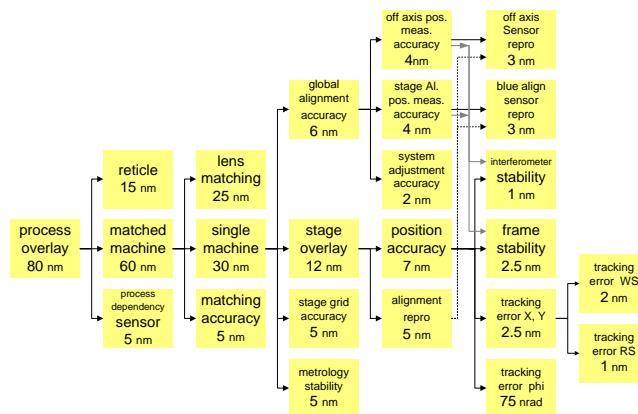


Figure 1.10: Example of a quantified understanding of overlay in a wafer stepper

Figure 1.10 shows a graphical example of an “overlay” budget for a wafer stepper. This figure is taken from the *System Design Specification* of the ASML TwinScan system, although for confidentiality reasons some minor modifications have been applied. This budget is based on a model of the overlay functionality in the wafer stepper. The budget is used to provide requirements for subsystems

and components. The actual contributions to the overlay are measured during the design and integration process, on functional models or prototypes. These measurements provide early feedback of the overlay design. If needed the budget or the design is changed on the basis of this feedback.

## 1.5 Coping with uncertainty

The architect has to make decisions all the time, while most substantiating data is still missing. On top of that some of the available data will be false, inconsistent or interpreted wrong.

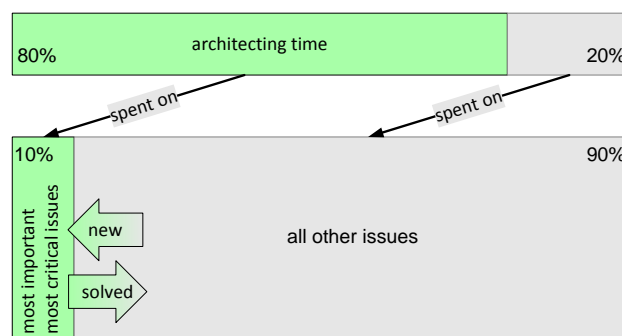


Figure 1.11: The architect focuses on important and critical issues, while monitoring the other issues

An important means in making decisions is building up insight, understanding and overview, by means of structuring the problems. The understanding is used to determine important (for the product use) and critical (with respect to technical design and implementation) issues. The architect will pay most attention to these *important* and *critical* issues. The other issues are monitored, because sometimes minor details turn out to be important or critical issues. Figure 1.11 visualizes the time distribution of the architect: 80% of the time is spent on 10% of the issues.

The architect will, often implicitly, work on the basis of a top 10 issue list, the ten most relevant (important, urgent, critical) issues. Figure 1.12 shows an example of such a “worry”-list.

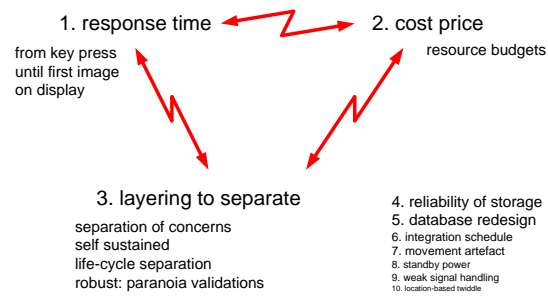


Figure 1.12: Example worry list of an architect

## 1.6 Modelling

Modelling is one of the most fundamental tools of an architect.

A **model** is  
a **simplified** representation of  
*part* of the **real world** used for:

communication, documentation  
analysis, simulation,  
decision making, verification

In summary models are used to obtain insight and understanding, facilitating communication, documentation, analysis, simulation, decision making, verification. At the same time the architect is always aware of the (over)simplification applied in every model. A model is very valuable, but every model has its limitations, imposed by the simplifications.

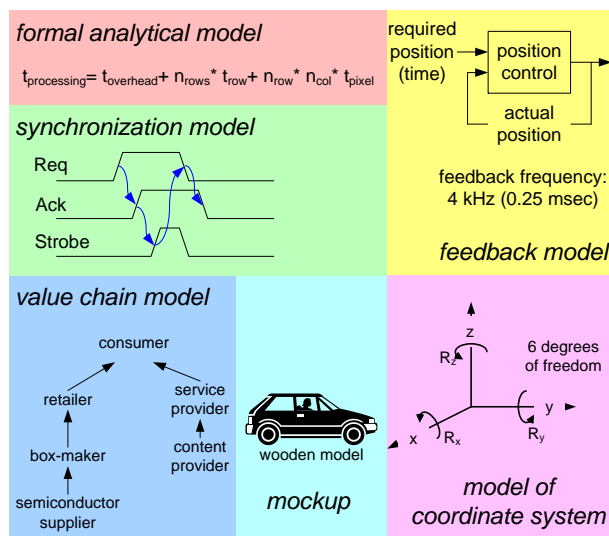


Figure 1.13: Some examples of models

Models exist in a very rich variety, an arbitrary small selection of models is shown in figure 1.13.

Models have many different manifestations. Figure 1.14 shows some of the different types of models, expressed in a number of adjectives.

Models can be *mathematical*, expressed in formulas, they can be *linguistic*, expressed in words or they can be *visual*, captured in diagrams. A model can be

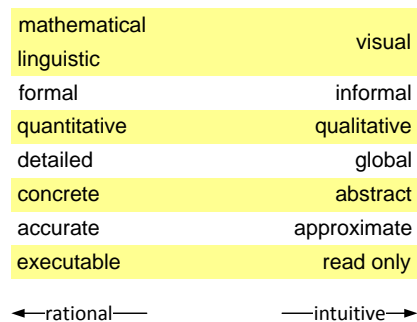


Figure 1.14: Types of models

formal, where notations, operations and terms are precisely defined, or informal using plain English and sketches. Quantitative models use meaningful numbers, allowing verification and judgements. Qualitative models show relations and behavior, providing understanding. Concrete models use tangible objects and parameters, while abstract models express mental concepts. Some models can be executed (as a simulation), while other models only make sense for humans reading the model.

## 1.7 WWHWWW questions

Why	Who
What	When
How	Where

Figure 1.15: The starting words for questions by the architect

All “W” questions are an important tool for the architect. Figure 1.15 shows the useful starting words for questions to be asked by an architect.

**Why**, **what** and **how** are used over and over in architecting. Why, what and how are used to determine objectives, rationale and design. This works highly recursively, a design has objectives and a rationale and results in smaller designs that again have objectives and rationales. Figure 1.16 shows that the recursion with **why** questions broadens the scope, and recursion with **how** questions opens more details in a smaller scope.

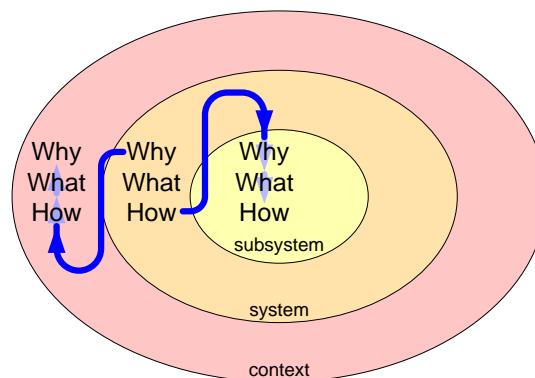


Figure 1.16: Why broadens scope, How opens details

**Who**, **where** and **when** are used somewhat less frequently. Who, where and when can be used to build up understanding of the context, and are used in cooperation with the project leader to prepare the project plan.

## 1.8 Decision Making Approach in Specification and Design

Many specification and design decisions have to be taken during the product creation process. For example, functionality and performance requirements need to be defined, and the way to realize them has to be chosen. Many of these decisions are interrelated and have to be taken at a time when many uncertainties still exist.

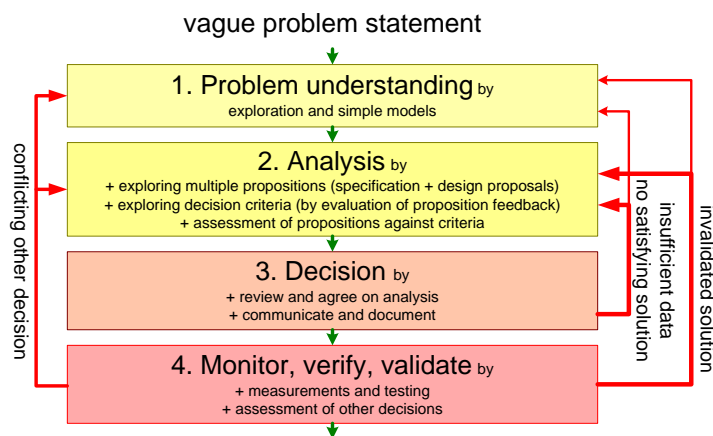


Figure 1.17: Flow from problem to solution

An approach to make these decisions is the flow depicted in Figure 1.17. The decision process is modeled in four steps. An understanding of the problem is created by the first step *problem understanding*, by exploration of problem and solution space. Simple models, in problem space as well as in solution space, help to create this understanding. The next step is to perform a somewhat more systematic *analysis*. The analysis is often based on *exploring multiple propositions*. The third step is the *decision* itself. The analysis results are reviewed, and the decision is documented and communicated. The last step is to *monitor, verify and validate* the decision.

The *analysis* involves multiple substeps: *exploring multiple propositions*, *exploring decision criteria* and *assessing the propositions against the criteria*. A proposition describes both specification (**what**) and design (*how*). Figure 1.18 shows an example of multiple propositions. In this example a high performance, but high cost alternative, is put besides two lower performing alternatives. Most criteria get articulated in the discussions about the propositions: “I think that we should choose proposition 2, because...”. The *because* can be reconstructed into a criterion.

The decision to chose a proposition is taken on the basis of the analysis results. A review of the analysis results ensures that these results are agreed upon. The

throughput	20 p/m	high-performance sensor	350 ns
cost	5 k\$	high-speed moves	9 m/s
safety		additional pipelining	
<i>low cost and performance 1</i>			
throughput	20 p/m	high-performance sensor	300 ns
cost	5 k\$	high-speed moves	10 m/s
safety			
<i>low cost and performance 2</i>			
throughput	25 p/m	highperformance sensor	200 ns
cost	7 k\$	high-speed moves	12 m/s
safety		additional collision detector	
<i>high cost and performance</i>			

Figure 1.18: Multiple propositions

decision itself is documented and communicated<sup>2</sup>. In case of insufficient data or in absence of a satisfying solution we have to back track to the *analysis* step. Sometimes it is better to revisit the problem statement by going back to the *understanding* step.

Taking a decision requires a lot of follow up. The decision is in practice based on partial and uncertain data, and many assumptions. A significant amount of work is to monitor the consequences and implementation of the decision. Monitoring is partially a *soft skill*, such as actively listening to engineers, and partially a *engineering activity* such as measuring and testing. The consequence of a measurement can be that the problem has to be revisited, starting again with the understanding for serious mismatches (“apparently we don’t understand the problem at all”) or direct into the analysis for smaller mismatches.

The implementation of taken decisions can be disturbed by later decisions. This problem is partially tackled by requirements traceability, where known interdependencies are managed explicitly. In the complex real world the amount of dependencies is almost infinite, that means that the explicit dependability specifications are inherently incomplete and only partially understood. To cope with the inherent uncertainty about dependabilities, an open mind is needed when screening later decisions. A conflict caused by a later decision triggers a revisit of the original problem.

The same flow of activities is used recursively at different levels of detail, as shown in Figure 1.19. A *system* problem will result in a system design, where many design aspects need the same flow of problem solving activities for the subsystems. This process is repeated for smaller scopes until termination at problems that can be solved directly by an implementation team. The smallest scope of termination is denoted as *atomic* level in the figure. Note that the more detailed problem solving might have impact on the more global decisions.

<sup>2</sup>This sounds absolutely trivial, but unfortunately this step is performed quite poorly in practice.

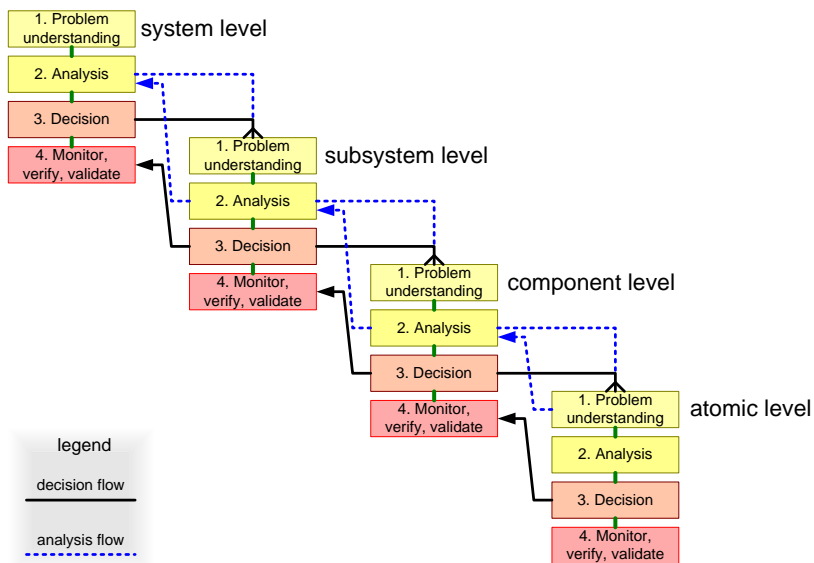


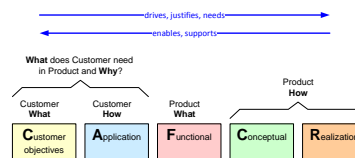
Figure 1.19: Recursive and concurrent application of flow

## 1.9 Acknowledgements

The team of composable architectures, with the following members Pierre America, Marcel Bijsterveld, Peter van den Hamer, Jürgen Müller, Henk Obbink, Rob van Ommering, and William van der Sterren within Philips Research provided valuable feedback for this article

## Chapter 2

# Short introduction to basic “CAFRCR” model



### 2.1 Introduction

A simple reference model is used to enable the understanding of the inside of a system in relation to its context.

An early tutorial[4] of this model used the concatenation of the first letters of the views in this model to form the acronym “CAFRCR” (Customer Objectives, Application, Functional, Conceptual, Realization). This acronym is used so often within the company, that changing the acronym has become impossible. We keep the name constant, despite the fact that better names for some of the views have been proposed. The weakest name of the views is *Functional*, because this view also contains the so-called *non functional* requirements. A better name for this view is the Black-Box view, expressing the fact that the system is described from outside, without assumptions about the internals.

The model has been used effectively in a wide variety of applications, ranging from motor way management systems to component models for audio/video streaming. The model is not a silver bullet and should be applied only if it helps the design team.

## 2.2 The CAFCR model

A useful top level decomposition of an architecture is provided by the so-called “CAFCR” model, as shown in figure 2.1. The *Customer Objectives* view and the *Application* view provide the **why** from the customer. The *Functional* view describes the **what** of the product, which includes (despite the name) also the *non-functional* requirements. The **how** of the product is described in the *Conceptual* and *Realization* view, where the conceptual view is changing less in time than the fast changing realization (Moore’s law!).

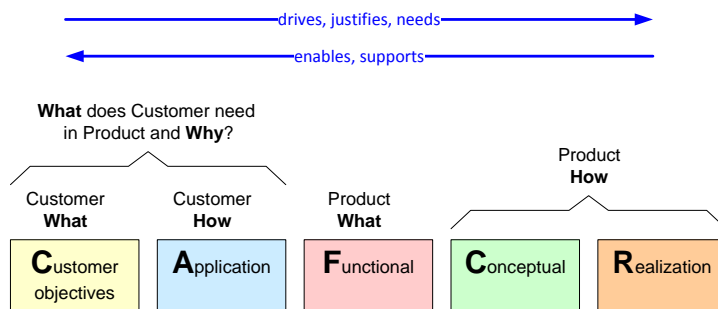


Figure 2.1: The “CAFCR” model

The job of the architect is to integrate these views in a consistent and balanced way. Architects do this job by *frequent viewpoint hopping*: looking at the problem from many different viewpoints, sampling the problem and solution space in order to build up an understanding of the business. Top-down (objective driven, based on intention and context understanding) in combination with bottom-up (constraint aware, identifying opportunities, know how based), see figure 2.2.

In other words the views must be used concurrently, not top down like the waterfall model. However at the end a consistent story-line must be available, where the justification and the needs are expressed at the customer side, while the technical solution side enables and support the customer side.

The model will be used to provide a next level of reference models and submethods. Although the 5 views are presented here as sharp disjunct views, many subsequent models and methods don’t fit entirely in one single view. This in itself not a problem, the model is a means to build up understanding, it is not a goal in itself.

## 2.3 Who is the customer?

The term *customer* is easily used, but it is far from trivial to determine the customer. The position in the value chain shows that multiple customers are involved. In

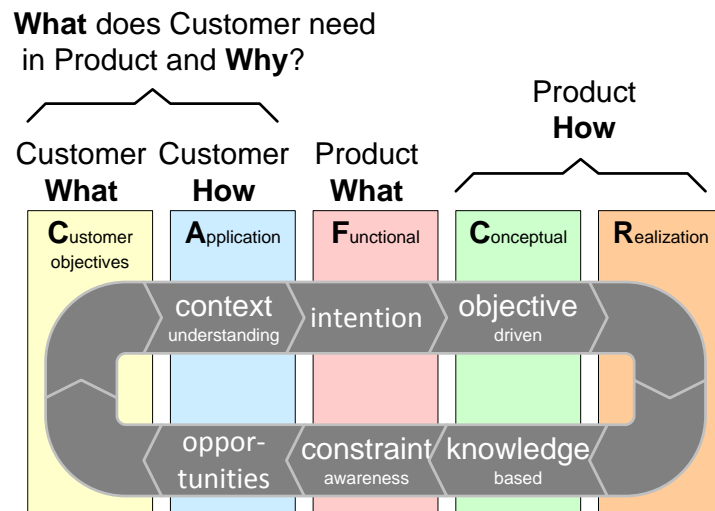


Figure 2.2: Five viewpoints for an architecture. The task of the architect is to integrate all these viewpoints, in order to get a *valuable, usable and feasible* product.

figure 2.3 the multiple customers are addressed by applying the CAFCR model recursively.

The customer is a gross generalization. Marketing managers make a classification of customers by means of a market segmentation. It is recommended to start building up insight by making specific choices for the customer, for example by selecting specific market segments. Making early assumptions about synergy between market segments can handicap the build-up of customer understanding. These kind of assumptions tend to pollute the model and inhibits clear and sharp reasoning.

many stakeholders are involved for any given customer. Multiple stakeholders are involved even when the customer is a consumer, such as parents, other family, and friends. Figure 2.4 shows an example of the people involved in a small company. Note that most of these people have different interests with respect to the system.

Market segments are also still tremendous abstractions. Architect have to stay aware all the time of the distance between the abstract models they are using and the reality, with all unique infinitely complex individuals.

## 2.4 Life Cycle view

The basic CAFCR model relates the customer needs to design decisions. However, in practice we have one more major input for the system requirements: the life

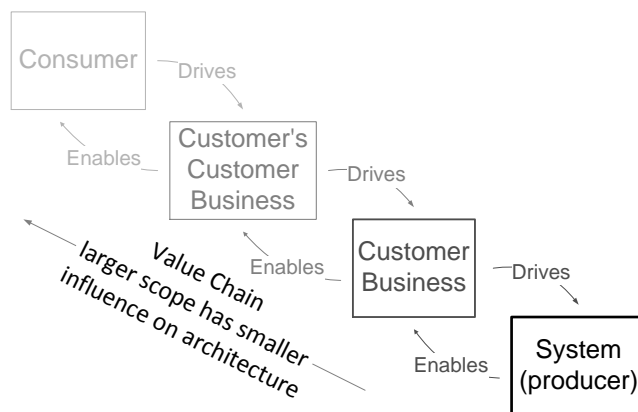


Figure 2.3: CAFCR can be applied recursively

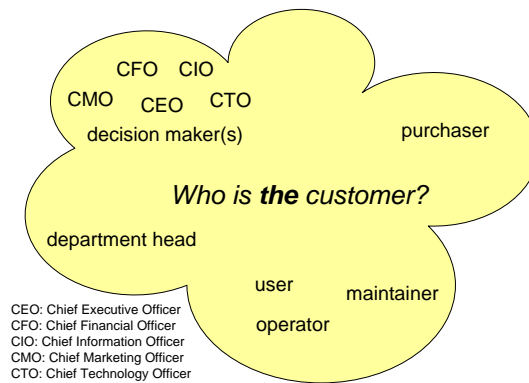


Figure 2.4: Which person is **the** customer?

cycle needs. Figure 2.5 shows the CAFCR+ model that extends the basic CAFCR model with a *Life Cycle view*.

The system life cycle starts with the conception of the system that trigger the development. When the system has been developed then it can be reproduced by manufacturing, ordered by logistics, installed by service engineers, sold by sales representatives, and supported throughout its life time. Once delivered every produced system goes through a life cycle of its own with scheduled maintenance, unscheduled repairs, upgrades, extensions, and operational support. Many stakeholders are involved in the entire life cycle: sales, service, logistics, production, R&D. Note that all these stakeholders can be part of the same company or that these functions can be distributed over several companies.

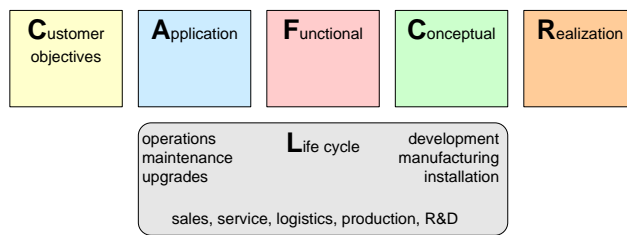
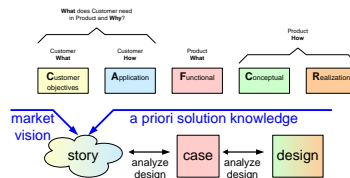


Figure 2.5: CAFCR+ model; Life Cycle View

# Chapter 3

## Story How To



### 3.1 Introduction

Starting a new product definition often derails in long discussions about generic specification and design issues. Due to lack of reality check these discussions are very risky, and often way too theoretical. Story telling followed by specific analysis and design work is a complementary method to do *in-depth* exploration of *parts* of the specification and design.

The method provided here, based on story telling, is a powerful means to get the product definition quickly in a concrete factual discussion. The method is especially good in improving the communication between the different stakeholders. This communication is tuned to the stakeholders involved in the different CAFCR views: the *story* and *use case* can be exchanged in ways that are understandable for both marketing-oriented people as well as for designers.

Figure 3.1 positions the story in the customer objectives view and application view. A good story combines a clear market vision with a priori realization know how. The story itself must be expressed entirely in customer terms, no solution jargon is allowed.

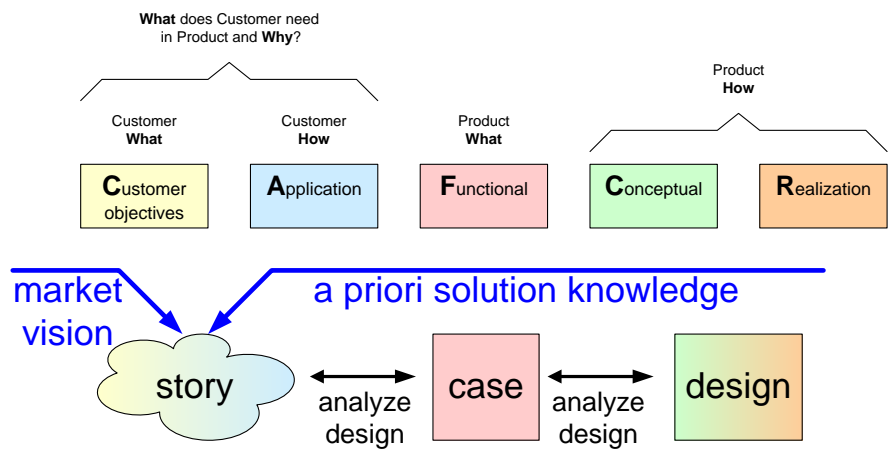


Figure 3.1: From story to design

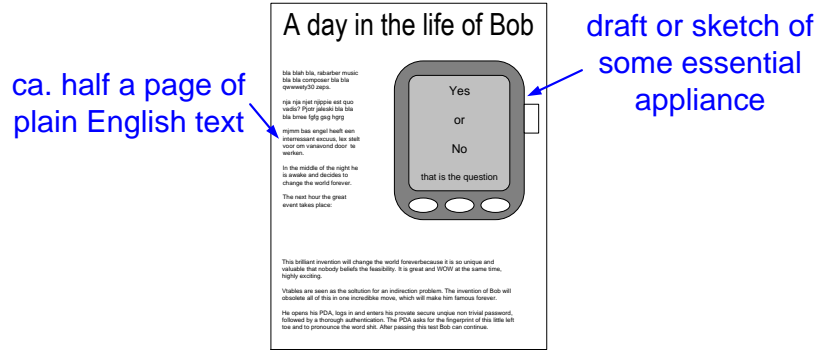


Figure 3.2: Example story layout

### 3.2 How to Create a Story?

A story is a short single page story, as shown in Figure 3.2, preferably illustrated with sketches of the most relevant elements of the story, for instance the look and feel of the system being used. Other media such as cartoons, animations, video or demonstrations using mockups can be used also. The *duration* or the *size* of the “story” must be limited to enable focus on the essentials.

Every story has a *purpose*, something the design team wants to learn or explore. The purpose of the story is often in the conceptual and realization views. The *scope* of the story must be chosen carefully. A wide scope is useful to understand a wide context, but leaves many details unexplored. An approach is to use recursively refined stories: an overall story setting the context and a few other stories zooming

in on aspects of the overall story.

The story can be written from several *stakeholder viewpoints*. The viewpoints should be carefully chosen. Note that the story is also an important means of communication with customers, marketing managers and other domain experts. Some of the stakeholder viewpoints are especially useful in this communication.

The *size* of the story is rather critical. Only short stories serve the purpose of discussion catalyst. At the same time all stakeholders have plenty of questions that can be answered by extending the story. It is recommended to really limit the size of the story. One way of doing this is by consolidating additional information in a separate document. For instance, in such a document the point of the story in customer perspective, the purpose of the story in the technology exploration, and the implicit assumptions about the customer and system context can be documented.

### 3.3 How to Use a Story?

The story itself must be very accessible for all stakeholders. The story must be attractive and appealing to facilitate communication and discussion between those stakeholders. The story is also used as input for a more systematic analysis of the product specification in the functional view. All functions, performance figures and quality attributes are extracted from the story. The analysis results are used to explore the design options.

Normally several iterations will take place between story, case and design exploration. During the first iteration many questions will be raised in the case analysis and design, which are caused by the story being insufficiently specific. This needs to be addressed by making the story more explicit. Care should be taken that the story stays in the Customers views and that the story is not extended too much. The story should be sharpened, in other words made more explicit, to answer the questions.

After a few iterations a clear integral overview and understanding emerges for this very specific story. This insight is used as a starting point to create a more complete specification and design.

### 3.4 Criteria

Figure 3.3 shows the criteria for a good story. It is recommended to assess a story against this checklist and either improve a story such that it meets all the criteria or to reject the story. Fulfillment of these criteria helps to obtain a useful story. The set of five criteria is a necessary but not sufficient set of criteria. The value of a story can only be measured in retrospect by determining the contribution of the story to the specification and design process.

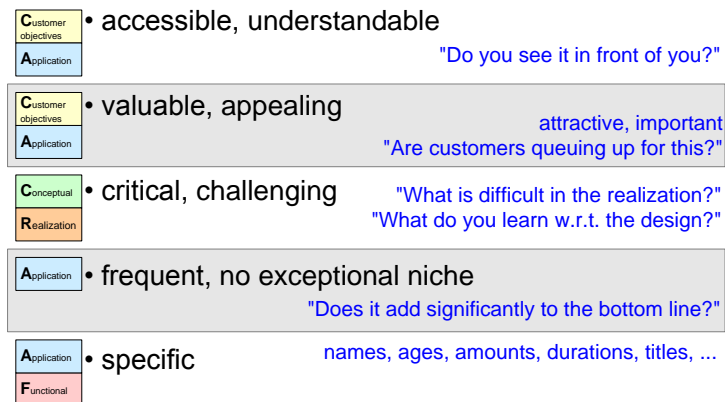


Figure 3.3: criteria for a good story

**Accessible, understandable** The main function of a story is to make the opportunity or problem communicable with all the stakeholders. This means that the story must be accessible and understandable for all stakeholders. The description or presentation should be such that all stakeholders can *live through*, *experience* or *imagine* the story. A “good” story is not a sheet of paper, it is a living story.

**Important, valuable, appealing, attractive** The opportunity or problem (idea, product, function or feature) must be significant for the target customers. This means that it should be important for them, or valuable; it should be appealing and attractive.

Most stories fail on this criterium. Some so-so opportunity (whistle and bell-type) is used, where nobody gets really enthusiastic. If this is the case more creativity is required to change the story to an useful level of importance.

**Critical, challenging** The purpose of the story is to learn, define, analyze new products or features. If the implementation of a story is trivial, nothing will be learned. If all other criteria are met and no product exists yet, than just do it, because it is clearly a quick win!

If the implementation is challenging, then the story is a good vehicle to study the trade-offs and choices to be made.

**Frequent, no exceptional niche** Especially in the early exploration it is important to focus on the main line, the *typical* case. Later in the system design more specialized cases will be needed to analyze for instance more exceptional worst case situations.

A *typical* case is characterized by being frequent, it should not be an exceptional niche.

**Specific** The value of a story is the specificity. Most system descriptions are very generic and therefore very powerful, but at the same time very non specific. A good story provides focus on a single story, one occasion only. In other words the thread of the story should be very specific.

Specificity can be achieved in social, cultural, emotional or demographic details, such as names, ages, and locations. “Eleven year old Jane in Shanghai” is a very different setting than “Eighty two year old John in an Amsterdam care center”. Note that these social, cultural, emotional or demographic details also help in the engagement of the audience. More analytical stories can be too “sterile” for the audience.

Another form of specificity is information that helps to quantify. For example, using “Doctor Zhivago” as movie content sets the duration to 200 minutes. Stories often need lots of these kinds of detail to facilitate later specification and design analysis. When during the use of the story more quantification is needed, then the story can be modified such that it provides that information.

A good story is in **all** aspects as specific as possible, which means that:

- persons playing a role in the story preferably have a name, age, and other relevant attributes
- the time and location are specific (if relevant)
- the content is specific (for instance is listening for **2 hours** to songs of **the Beatles**)

Story writers sometimes want to show multiple possibilities and describe somewhere an escaping paragraph to fit in all the potential goodies (Aardvark works, sleeps, eats, swims et cetera, while listening to his Wow56). Simply leave out such a paragraph, it only degrades the focus and value of the story.

### 3.5 Example Story

Figure 3.4 shows an example of a story for hearing aids. The story first discusses the problem an elderly lady suffers from due to imperfect hearing aids. The story continues with postulated new devices that helps her to participate again in an active social life.


Figure 3.5 shows for the value and the challenge criteria what this story contributes.

Betty is a 70-year-old woman who lives in Eindhoven. Three years ago her husband passed away, and since then, she lives in a home for the elderly. Her two children, Angela and Robert, come and visit her every weekend, often with Betty's grandchildren Ashley and Christopher. As with so many women of her age, Betty is reluctant to touch anything that has a technical appearance. She knows how to operate her television, but a VCR or even a DVD player is way to complex.

When Betty turned 60, she stopped working in a sewing studio. Her work in this noisy environment made her hard-of-hearing with a hearing-loss of 70dB around 2kHz. The rest of the frequency spectrum shows a loss of about 45dB. This is why she had problems understanding her grandchildren and why her children urged her to apply for hearing aids two years ago. Her technophobia (and her first hints or arthritis) inhibit her from changing her hearing aids' batteries. Fortunately, her children can do this every weekend.

This Wednesday, Betty visits the weekly Bingo afternoon in the meeting place of the old-folk's home. It's summer now and the tables are outside. With all those people there, it's a lot of chatter and babble. Two years ago, Betty would never go to the bingo: "I cannot hear a thing when everyone babbles and clatters with the coffee cups. How can I hear the winning numbers?!". Now that she has her new digital hearing instruments, even in the bingo cacophony, she can understand everyone she looks at. Her social life has improved a lot, and she even won the bingo a few times.

That same night, together with her friend Janet, she attends Mozart's opera The Magic Flute. Two years earlier, this would have been one big low rumbly mess, but now she even hears the sparkling high piccolos. Her other friend Carol never joins their visits to the theaters. Carol also has hearing aids; however, hers only "work well" in normal conversations. "When I hear music, it's as if a butcher's knife cuts through my head. It's way too sharp!". So Carol prefers to take her hearing aids out, missing most of the fun. Betty is so happy that her hearing instruments simply know where they are and adapt to their environment.

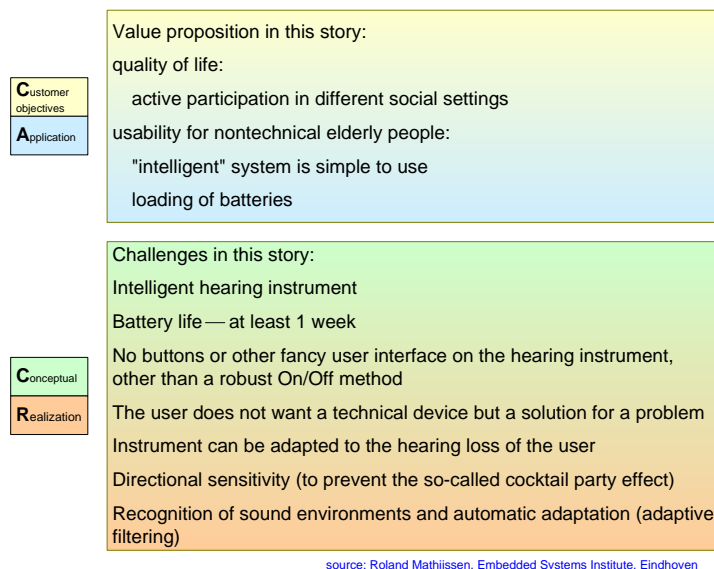


source: Roland Mathijssen  
Embedded Systems Institute  
Eindhoven

Figure 3.4: Example of a story

### 3.6 Acknowledgements

Within the IST-SWA research group lots of work has been done on scenario and story based architecting, amongst others by Christian Huiban and Henk Obbink. Rik Willems helped me to sharpen the *specificity* criterium. Melvin Zaya provided feedback on the importance of the story context and the "point" of the story. Roland Mathijssen provided an example story.



source: Roland Mathijssen, Embedded Systems Institute, Eindhoven

Figure 3.5: Value and Challenges in this story

# Bibliography

- [1] Gerrit Muller. The system architecture homepage. <http://www.gaudisite.nl/index.html>, 1999.
- [2] Gerrit Muller. The role and task of the system architect. <http://www.gaudisite.nl/RoleSystemArchitectPaper.pdf>, 2000.
- [3] Gerrit Muller. Function profiles; the sheep with 7 legs. <http://www.gaudisite.nl/FunctionProfilesPaper.pdf>, 2001.
- [4] Henk Obbink, Jürgen Müller, Pierre America, and Rob van Ommering. COPA: A component-oriented platform architecting method for families of software-intensive electronic products. [http://www.hitech-projects.com/SAE/COPA/COPA\\_Tutorial.pdf](http://www.hitech-projects.com/SAE/COPA/COPA_Tutorial.pdf), 2000.
- [5] William H. Press, William T. Vetterling, Saul A. Teulosky, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, 1992. Simulated annealing methods page 444 and further.
- [6] Eberhardt Rechtin and Mark W. Maier. *The Art of Systems Architecting*. CRC Press, Boca Raton, Florida, 1997.

## History

**Version: 1.4, date: September 25, 2014 changed by: Gerrit Muller**

- added summary

**Version: 1.3, date: November 29, 2010 changed by: Gerrit Muller**

- removed CAFCR (too much duplicates)

**Version: 1.2, date: July 6, 2004 changed by: Gerrit Muller**

- removed FCR views, and qualities

**Version: 1.1, date: March 24, 2004 changed by: Gerrit Muller**

- created reader

**Version: 1.0, date: June 20, 2002 changed by: Gerrit Muller**

- Added Basic working method architect
- Added Basic CAFCR
- Added CAFCR views

- Added Qualities
- Version: 0.1, date: February 20, 2002 changed by: Gerrit Muller**
- Added "Scenario How To"
  - changed the exercise