

Module System Architect Toolkit



Gerrit Muller

Embedded Systems Institute

Den Dolech 2 (Laplace Building 0.10) P.O. Box 513, 5600 MB Eindhoven The Netherlands

`gerrit.muller@embeddedsystems.nl`

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.

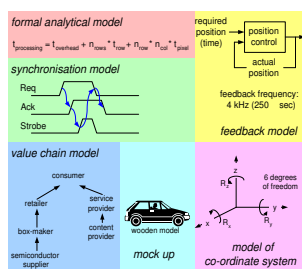
All Gaudí documents are available at:
<http://www.gaudisite.nl/>

Contents

1	Basic Working Methods of a System Architect	1
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	11
1.7	WWHWWW	13
1.8	Problem solving approach	14
1.9	Acknowledgements	16
2	Short introduction to basic “CAFCR” model	18
2.1	Introduction	18
2.2	The CAFCR model	18
2.3	Who is the customer?	19
2.4	Life Cycle view	20
3	Story How To	23
3.1	Introduction	23
3.2	Criteria	25
3.2.1	Accessible, understandable	26
3.2.2	Important, valuable, appealing, attractive	26
3.2.3	Critical, challenging	26
3.2.4	Frequent, no exceptional niche	26
3.2.5	Specific	26
3.3	Acknowledgements	27

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

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.

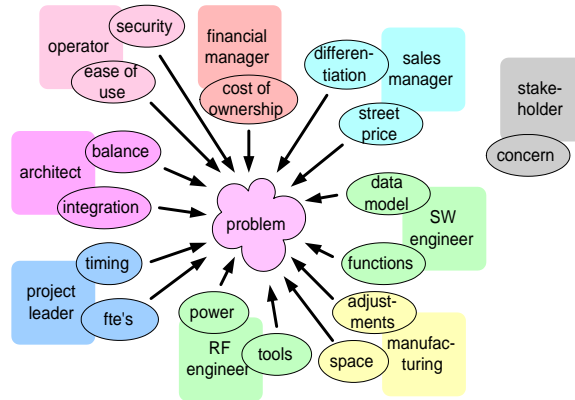


Figure 1.1: Small subset of viewpoints

Figure 1.2 shows an example of the successive viewpoints used by the architect. This sequence can take anywhere from minutes to weeks.

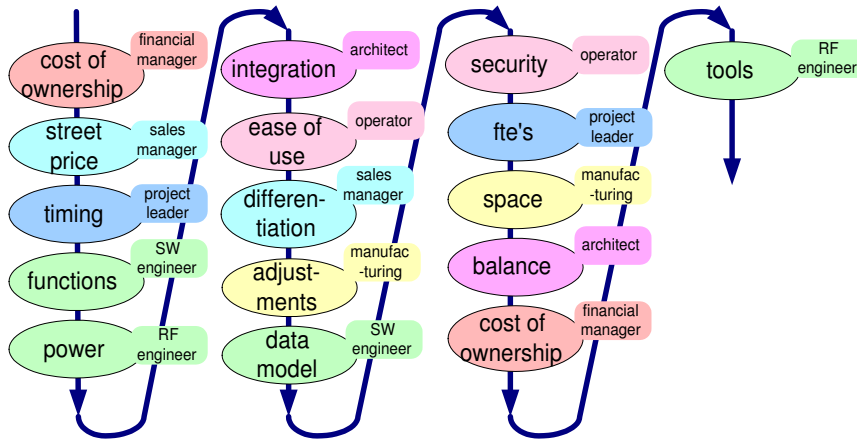


Figure 1.2: Viewpoint Hopping

This 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 his head) appears to be quite random. Figure 1.3 shows an example of an exploration path happening inside the architects head.

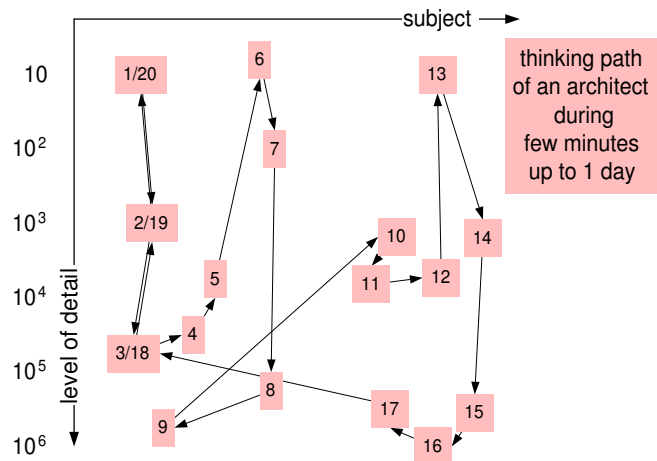


Figure 1.3: The seemingly random exploration path

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, which 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 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,

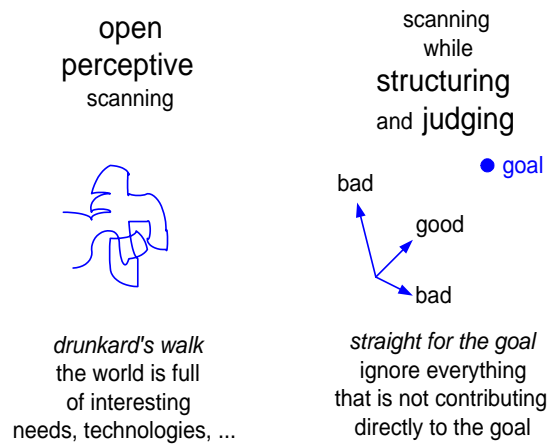


Figure 1.4: Two modes of scanning by an architect

leads to a local, but not necessarily a global, minimum. Nature's own minimization algorithm is based on a quite different procedure...

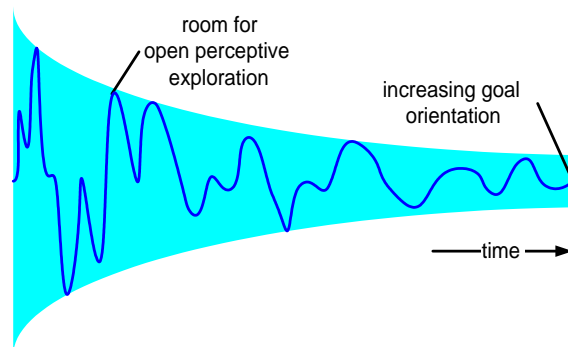


Figure 1.5: Combined open perceptive scanning and goal-oriented scanning

See also figure 1.5 for the combined scanning path.

The 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.

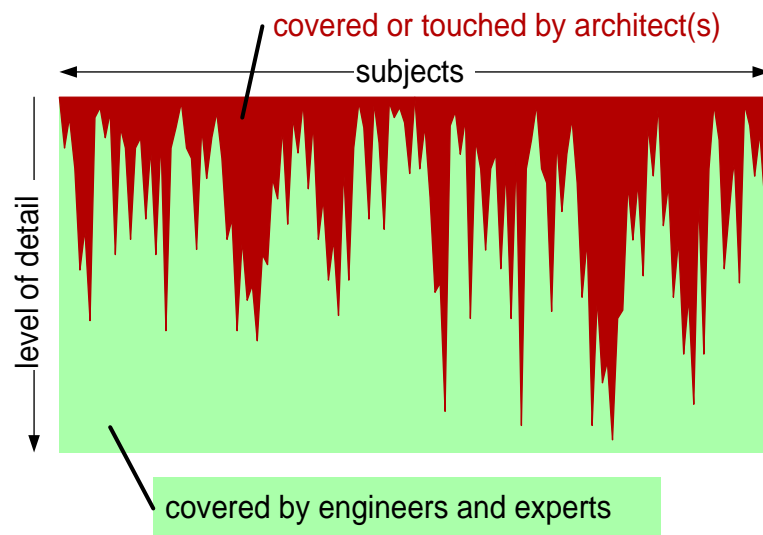


Figure 1.6: The final coverage of the problem and solution space by architect and engineers

1.3 Decomposition and integration

The architect applies a reduction strategy by means of decomposition over and over, as shown in figure 1.7. Decomposition is a very generic principle, which can be applied for many different problem and solution dimensions, as will be shown in the later sections.

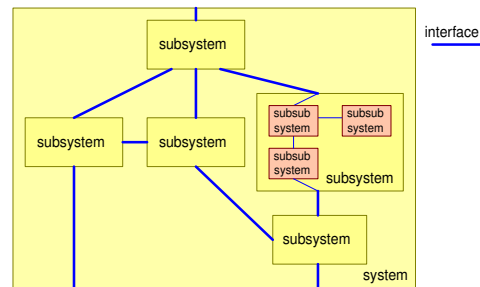


Figure 1.7: Decomposition, interface management and integration

Whenever something is decomposed the resulting components will be decoupled by interfaces. The architect will invest time in interfaces, since these provide a convenient method to determine system structure and behavior, while decoupling the inside of these components from their external behavior.

The true challenge for the architect is to design decompositions, which in the end will support an integration of components into a system. Most effort of the architect is concerned with the integrating concepts, how do multiple components work together?

Many stakeholders perceive the decomposition and the interface management as the most important contribution. The synthesis or integration part is more difficult and time consuming, and will be perceived as the main contribution by the architect himself.

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.

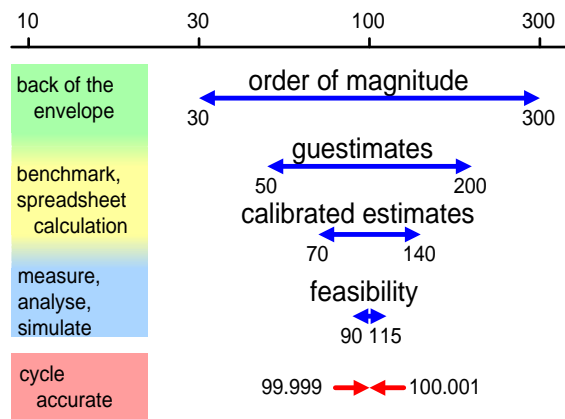


Figure 1.8: Successive quantification refined

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 which 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.

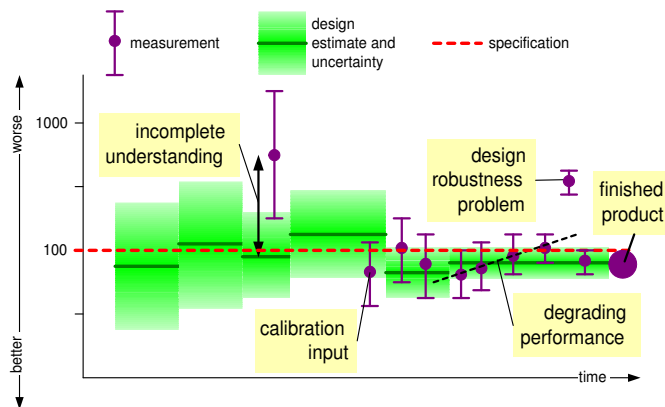


Figure 1.9: Example of the evolution of quantification in time

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 green area indicates the “paper” value, with its accuracy. In dark purple the measurements are shown. 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).

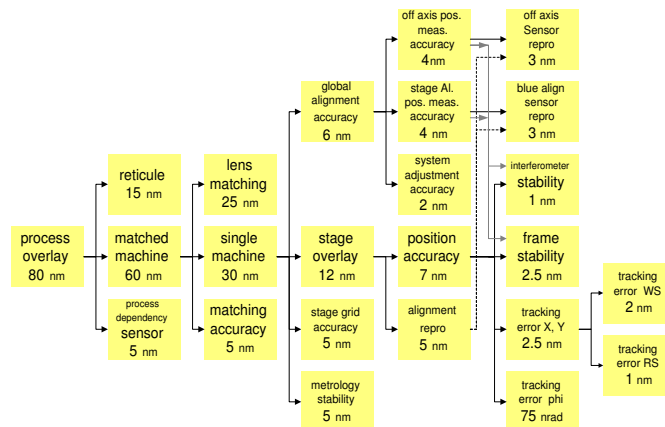


Figure 1.10: Example of a quantified understanding of overlay in a waferstepper

Quantification is based on modelling. A quantified model can be used as budget, an instrument to guide the specification and design process and to monitor the status. Figure 1.10 shows a graphical example of a budget, for a waferstepper.

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.

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. These issues will get most of the architects attention. The other issues are monitored, sometimes minor details turn out to be important or critical issues. Figure 1.11 visualizes the time distribution of the architect.

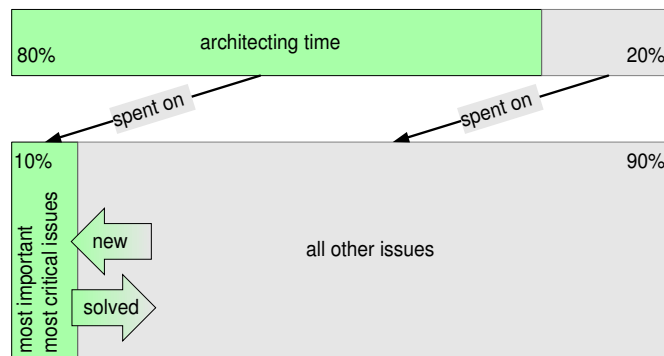


Figure 1.11: The architect focuses on important and critical issues, while monitoring the other issues

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.

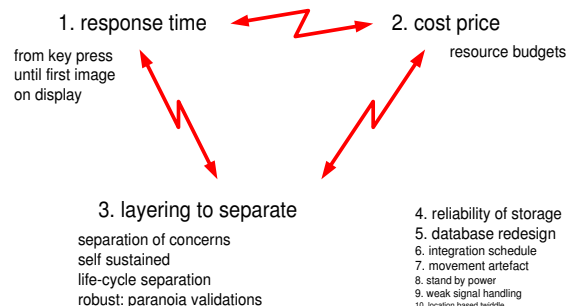


Figure 1.12: Example worry list of an architect

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, which enable all of its purposes. 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.

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

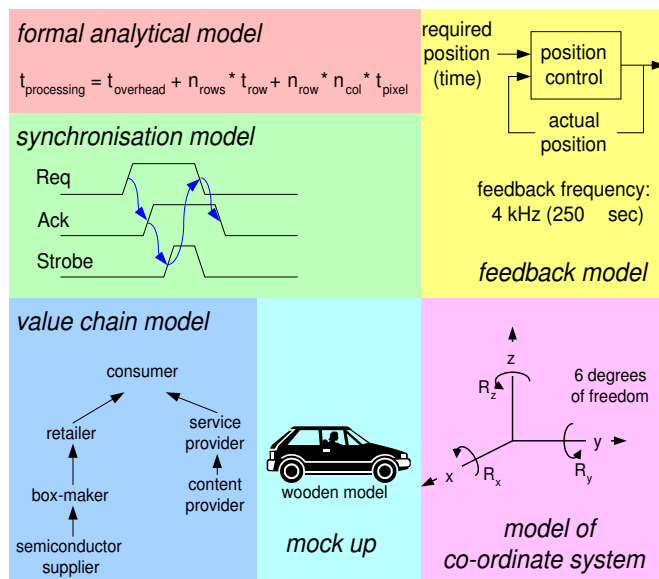


Figure 1.13: Some examples of models

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, textual, expressed in words or visual, visualized in diagrams. A model can be formal, where notations, operations and terms are precisely defined, or informal using plain English and sketches.

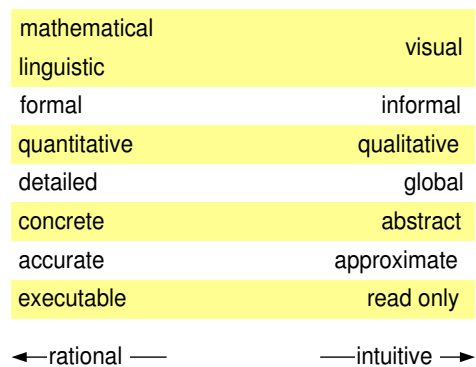


Figure 1.14: Types of models

Quantitative models use meaningful numbers, allowing verification and judgments; 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

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

Why	Who
What	When
How	Where

Figure 1.15: The starting words for questions by the 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 recursive, a design has objectives and a rationale and results in smaller designs which again have objectives and rationales.

Who, where and when are somewhat less frequently used. 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.

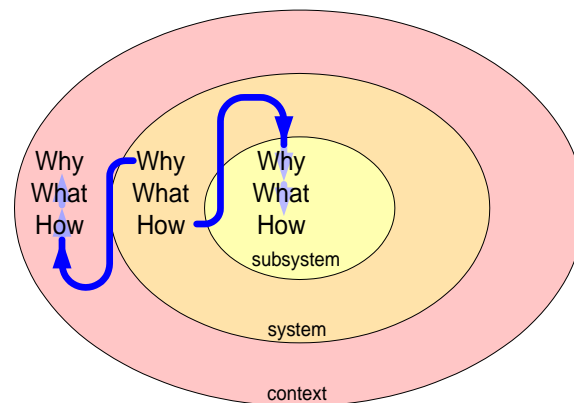


Figure 1.16: Why broadens scope, How opens details

1.8 Problem solving approach

The design process follows the flow as depicted in figure 1.17. The first step is *problem understanding*, by exploration of problem and solution space an understanding of the problem is created. Simple models, in problem as well as in solution space help to create this understanding.

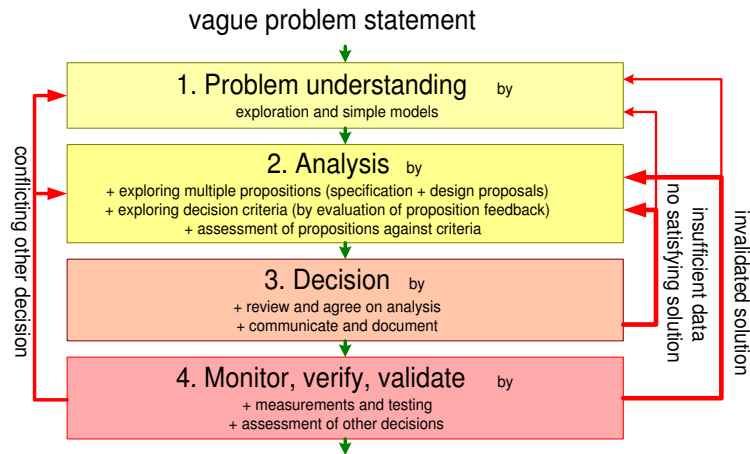


Figure 1.17: Flow from problem to solution

The next step is to perform a somewhat more systematic *analysis*. This step involves multiple substeps: *exploring multiple propositions*, where a proposition is a combined specification and design. Figure 1.18 shows an example of multiple propositions. In this example a high performance, but high cost alternative is put beside two lower performance alternatives.

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	high performance 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

The propositions are evaluated against criterions. Another substep in the *analysis*

is *the exploration of the decision criteria*. 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.

criteria	criteron weight	low cost and performance 1	low cost and performance 2	high cost and performance
throughput	5	2	2	3
cost	5	3	3	2
safety	5	5	5	5
future proof	2	2	3	3
effort	4	5	4	4
dev. time	5	5	4	4
risk	4	4	3	3
maintenance	3	2	3	3

Figure 1.19: Assessment of propositions

The analysis results in an evaluation of the propositions by means of the criteria, for instance as shown in figure 1.19.

The analysis results needs to be agreed upon and to be reviewed in order to take a decision about the proposition to be chosen. This decision needs to be communicated and documented¹.

Taking a decision requires a lot of follow up. The decision is in practice based on partial and uncertain data, and many assumptions. An 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.

Another source of disturbance with respect to taken decisions are new decisions which might have impact on the decisions already known. This problem is partially tackled by requirements traceability, where known interdependencies are managed explicitly. In the complex real world the amount of dependencies is near infinite, which means that the explicit dependability specifications are inherently incomplete and only partially understood.

To cope with the inherent uncertainty about dependabilities a continuous open mind is needed when screening later decisions. The original problem might have to be revisited, due to a conflict with another later decision.

The same flow of activities is used recursively at different levels of details. A “global” problem will result in a high level design, where many design aspects

¹This sounds absolutely trivial, but unfortunately this step is quite poor performed in practice.

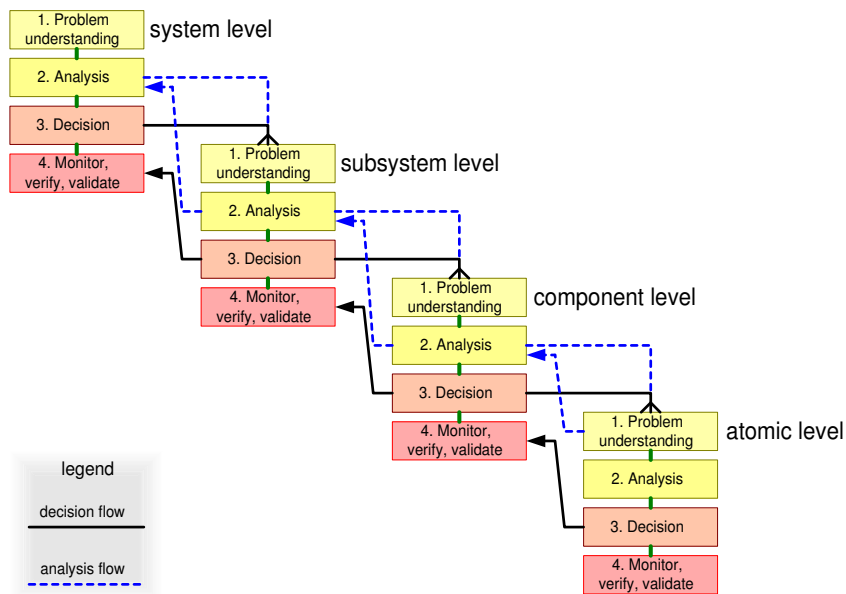


Figure 1.20: Recursive and concurrent application of flow

need the same flow of problem solving activities. Figure 1.20 shows the recursive application of this approach. Note that the more detailed problem solving might have impact on the more global decisions.

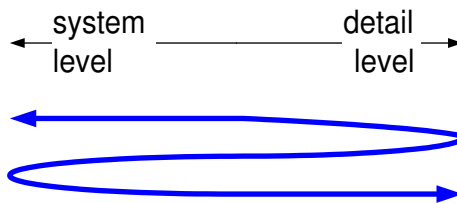


Figure 1.21: Exploration by rapid iteration

The approach should not be applied in a strict top down fashion. Often more detailed understanding is required to take a high level decision. Figure 1.21 shows that exploration is performed by fast iteration from system level to detail level and back.

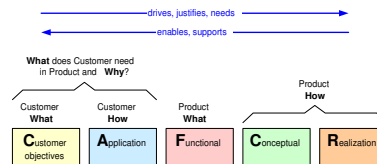
1.9 Acknowledgements

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 “CAFPCR” 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 “CAFPCR” (Customer Objectives, Application, Functional, Conceptual, Realization). This acronym is used so often within the company, that changing the acronym has become impossible. Although better names for some of the views have been proposed, the names are not changed. The weakest name of the views is *Functional*, because this view also contains the so-called *non functional* requirements.

The model has been used effectively in a wide variety of applications, ranging from motorway 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 CAFPCR model

A useful top level decomposition of an architecture is provided by the so-called “CAFPCR” 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!).

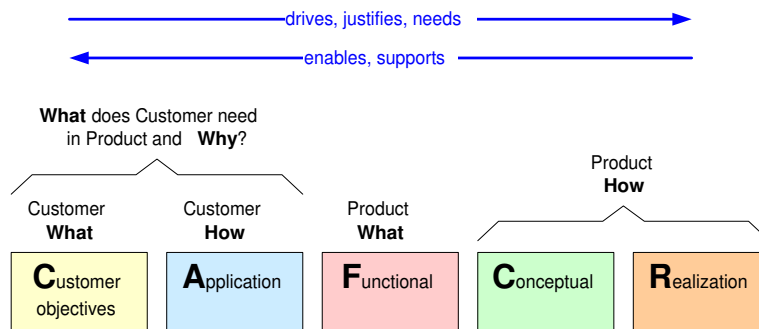


Figure 2.1: The “CAFCR” model

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 must be available, where the justification and the needs are expressed in 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 methods. 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 2.3 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. Figure 2.4 shows a number of examples of market segmentation axis and related segments.

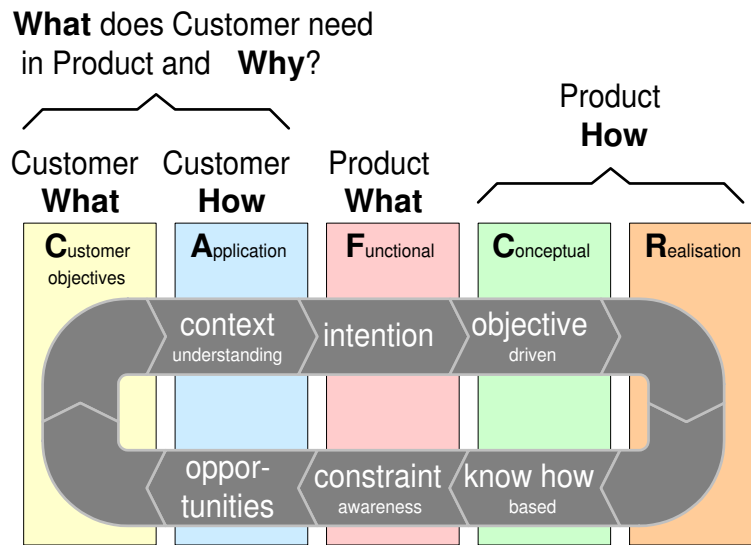


Figure 2.2: Five viewpoints for an architecture. The task of the architect is to integrate all these viewpoints, in order to get a *valuable, usable and feasible* product.

It is recommended to start building up insight by making specific choices for the customer. Sometimes the model has to be made several times for different customers. In the beginning it should be avoided to make a priori re-use assumptions between different market segments. These kind of assumptions pollute the model and inhibits clear and sharp reasoning.

Once we have determined the customer as a family or a company, it becomes clear that again multiple people are involved as customers. Figure 2.5 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.

Of course market segments are still tremendous abstractions. The architect has to stay aware all the time of the distance between the abstract models he is 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 cycle needs. Figure 2.6 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

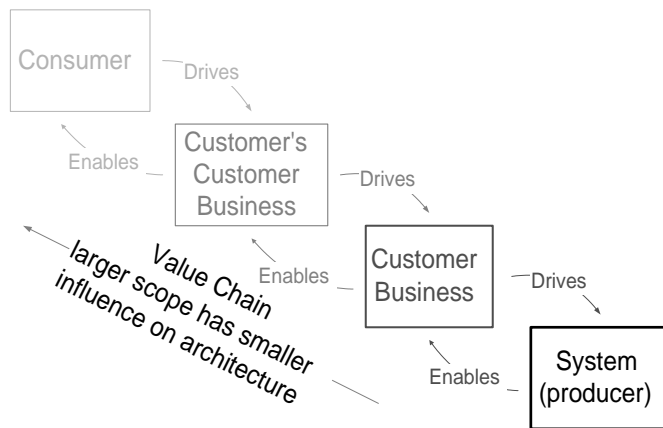


Figure 2.3: CAFCR can be applied recursively

segmentation axis	examples
geographical	USA, UK, Germany, Japan, China
business model	profit, non profit
economics	high end versus cost constrained
consumers	youth, elderly
outlet	retailer, provider, OEM, consumer direct

Figure 2.4: Market segmentation

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.

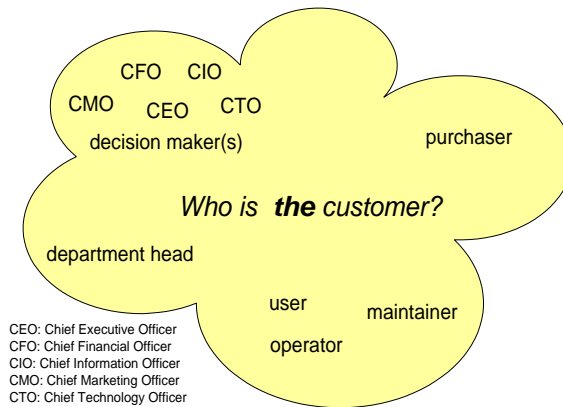


Figure 2.5: Which person is **the** customer?

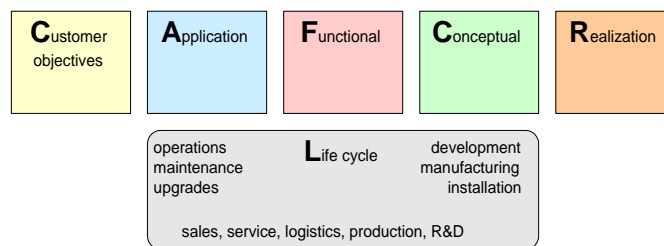
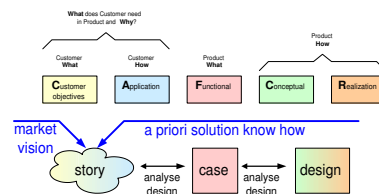


Figure 2.6: CAFCR+ model; Life Cycle View

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, way too academic. Once sufficient factual specification and design depth is obtained, it becomes time to determine useful generic concepts.

The method provided here, based on story telling, is a powerful means to get the product definition quickly in a concrete factual discussion.

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.

As shown in figure 3.2 a story is a short single page story, preferably illustrated with sketches of the most relevant elements of the story, for instance the appliance being used.

The story is used to get case data in the functional view. All functions, performance figures and quality attributes are extracted from the story. This case data is used to make a design exploration.

Normally several iterations will take place between story, case and design exploration. The first time many questions will be raised in the case analysis and design, which are addressed by making the story more explicit. Care should be

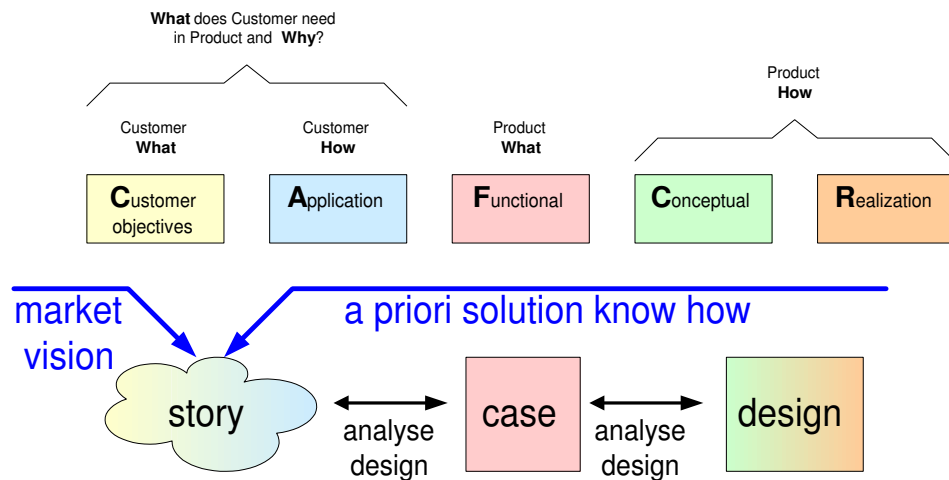


Figure 3.1: From story to design

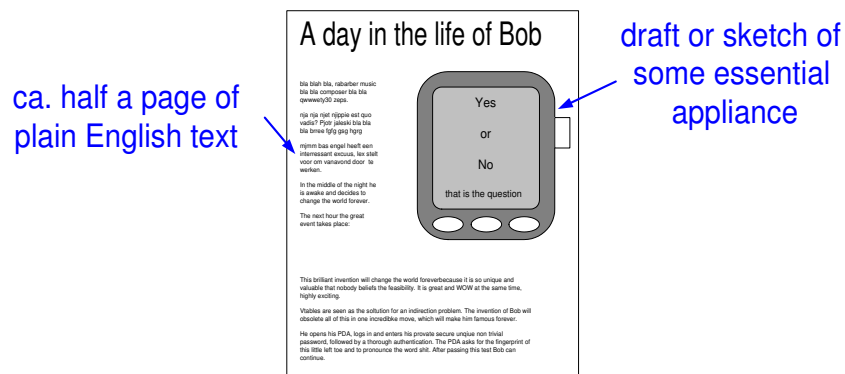


Figure 3.2: Example story layout

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.

Figure 3.3 shows a number of attentions points when writing a story. 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 good to understand a wide context, but leaves many details unexplored. A useful 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 points of view. The points of view should be carefully chosen. Note that the story is also an important means

- purpose
- scope
- viewpoint, stakeholders
- visualization
- size (max 1 A4)
- recursive decomposition, refinement

Figure 3.3: points of attention

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 catalysator. 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.

3.2 Criteria

Figure 3.4 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.

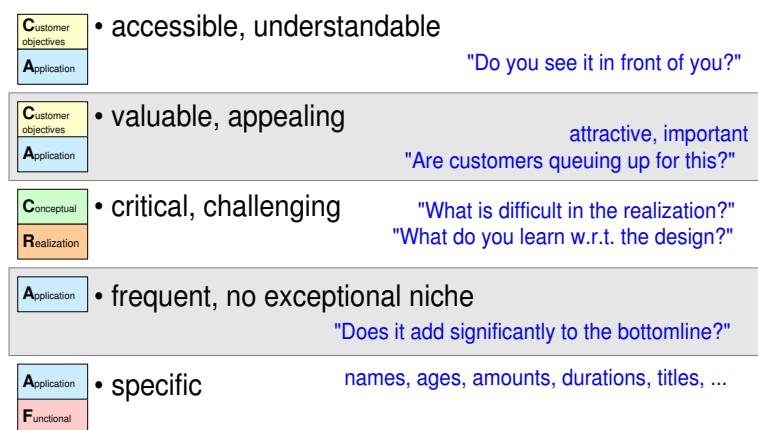


Figure 3.4: criteria for a good story

3.2.1 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 story is not a sheet of paper, it is a living story.

3.2.2 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 story's 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.

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

3.2.4 Frequent, no exceptional niche

Especially in the early exploration it is important to focus on the mainline, 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 is no exceptional niche which is described.

3.2.5 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.

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 an paragraph, it only degrades the focus and value of the story.

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**)

This kind of specific data is often needed to assess the other criteria, to bring it more alive, and in further analysis. If during the use of the story numbers have to be "invented", it is often best to improve the story by bringing in the numbers already in the story.

3.3 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.

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, 2000.
- [5] William H. Press, William T. Vetterling, Teulosky Saul A., 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: 1.2, 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