

# Human Measure and architecting

logo  
TBD

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 book bundles the human measure and architecting articles. The articles address the relationship between product creation and humans and the role of the system architect.

### **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>Introduction</b>	<b>ix</b>
<b>1 The System Architect; Meddler or Hero?</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Practical experience: memory usage in a medical workstation. . .	2
1.3 Introducing an Architect . . . . .	6
1.4 What is architecting? . . . . .	8
1.5 CBA: Conclusions, Benchmarking and Acknowledgements . . . .	13
1.5.1 Conclusions . . . . .	13
1.5.2 Benchmarking . . . . .	13
1.5.3 Acknowledgements . . . . .	14
1.5.4 Gaudí homepage . . . . .	15
1.6 Does architecture belong in the research laboratory? . . . . .	16
1.7 What is an architect? . . . . .	18
<b>2 The Importance of System Architecting for Development</b>	<b>20</b>
2.1 Introduction . . . . .	20
2.2 The Challenge . . . . .	21
2.3 Architecting . . . . .	23
2.4 Architecting Courses . . . . .	27
<b>3 Decomposing the Architect; What are Critical Success Factors?</b>	<b>31</b>
3.1 Introduction . . . . .	31
3.2 What is an Architect? . . . . .	32
3.3 Education . . . . .	35
3.4 Nature . . . . .	39
3.5 Experience . . . . .	42
3.6 Environment . . . . .	47
3.7 Discussion and Conclusions . . . . .	52
3.8 Acknowledgements . . . . .	53

<b>4</b>	<b>Architecting for Humans; How to Transfer Experience?</b>	<b>54</b>
4.1	Introduction . . . . .	54
4.2	User Experience . . . . .	57
4.3	Engineering . . . . .	62
4.4	Education . . . . .	64
4.5	Conclusion . . . . .	66
4.6	Acknowledgements . . . . .	67
<b>5</b>	<b>From the soft and fuzzy context to SMART engineering</b>	<b>68</b>
5.1	Introduction . . . . .	68
5.2	Case description . . . . .	70
5.3	Why SMART? . . . . .	72
5.4	Examples of smartening fuzzy requirements . . . . .	76
5.5	How to verify? . . . . .	80
5.6	Conclusion . . . . .	81
5.7	Acknowledgements . . . . .	82
<b>6</b>	<b>Communicating via CAFCR; illustrated by security example</b>	<b>83</b>
6.1	Introduction . . . . .	83
6.2	Stakeholders . . . . .	85
6.3	The "CAFCR" model and qualities . . . . .	87
6.4	Zooming in on security . . . . .	88
6.5	The wonder of communication . . . . .	92
6.6	Story telling . . . . .	94
6.7	Summary . . . . .	95
6.8	Acknowledgements . . . . .	96
<b>7</b>	<b>Architect and Human Measure; the integration role</b>	<b>97</b>
7.1	Introduction . . . . .	97
7.2	Illustration of the problem . . . . .	97
7.3	What is architecting? . . . . .	100
7.4	The architect . . . . .	102
<b>8</b>	<b>Architecture; the building as a product</b>	<b>104</b>
8.1	Introduction . . . . .	104
8.2	The Product: Building WDC . . . . .	104
8.3	The Product: Digital Video Recorder . . . . .	111
8.4	Discussion . . . . .	113
8.5	Conclusion . . . . .	114

# List of Figures

1.1	The position within the organisation . . . . .	2
1.2	Practical experience; a medical workstation . . . . .	3
1.3	Problem: unlimited memory consumption . . . . .	4
1.4	Solution: measure and redesign . . . . .	4
1.5	Method: process based budgeting . . . . .	5
1.6	Integration uncovers hidden problems . . . . .	5
1.7	Architecting scope . . . . .	6
1.8	Architecture awareness evolution . . . . .	7
1.9	The engineer's perception of the architect . . . . .	7
1.10	Architecting visualized . . . . .	8
1.11	Five viewpoints for an architecture. The task of the architect is to integrate all these viewpoints, in order to get a <i>valuable, usable</i> and <i>feasible</i> product. . . . .	9
1.12	The customer issues which are relevant for the case illustrated in this article . . . . .	10
1.13	Guiding <b>how</b> by providing five <i>how</i> -viewpoints . . . . .	11
1.14	Question generator . . . . .	11
1.15	Question generator . . . . .	12
1.16	The meddling architect as complementing factor in a team full of heroes . . . . .	13
1.17	Positioning the Gaudí research ambition in the worldwide efforts . . . . .	13
1.18	A small subset of contributors, here mostly participants of the System Architecture course . . . . .	14
1.19	Gaudí homepage . . . . .	15
1.20	Architecture and research? The technology management cycle . . . . .	16
1.21	Is architecting scientific? . . . . .	17
1.22	The architect as integrator . . . . .	18
1.23	The architect maintains technical roots . . . . .	19
2.1	The Challenge . . . . .	21
2.2	When all pieces fit ... . . . .	22
2.3	Simplified process view . . . . .	23

2.4	System architecture process . . . . .	23
2.5	What is architecting? . . . . .	24
2.6	"CAFCR" model . . . . .	24
2.7	Guiding how . . . . .	25
2.8	Gaudí ambition . . . . .	25
2.9	Operational hierarchy . . . . .	26
2.10	Architecting Scope . . . . .	26
2.11	Stakeholders Architect . . . . .	27
2.12	Profile of System Architect . . . . .	27
2.13	Draft System Architect Curriculum . . . . .	28
2.14	Course status in Philips . . . . .	28
2.15	Goals of 2-day Management SARCH course . . . . .	29
2.16	Program of 2-day Management SARCH . . . . .	29
3.1	Decomposing Contributing Factors . . . . .	32
3.2	Typical Development of a System Architect . . . . .	32
3.3	Growth in technical breadth, intermediate functions from specialist to system architect . . . . .	33
3.4	Different Architecting Scopes . . . . .	34
3.5	Proposed Curriculum for System Architects . . . . .	35
3.6	The outline of a CAFCR based architecting method . . . . .	35
3.7	Connecting System Design to Detailed Design . . . . .	36
3.8	Organizational Problem: Disconnect . . . . .	37
3.9	Architect: Connecting Problem and Technical Solution . . . . .	37
3.10	Major Bottleneck: Mental Dynamic Range . . . . .	38
3.11	Profile of an "Ideal" System Architect . . . . .	39
3.12	For Comparison: Profile of a Project Leader . . . . .	39
3.13	Project Leader versus System Architect . . . . .	40
3.14	Most Discriminating Characteristics . . . . .	40
3.15	Example: Trapezoid Pattern . . . . .	42
3.16	From SW input to physical Effect . . . . .	43
3.17	Discretization effects . . . . .	43
3.18	Example of Discretization Problem . . . . .	44
3.19	Example of Generic Smoothing Consideration . . . . .	45
3.20	Architects Collect a Rich Set of Patterns . . . . .	46
3.21	Simplified decomposition of the Business . . . . .	47
3.22	Line Organization Stovepipe . . . . .	48
3.23	Business Organization Stovepipe . . . . .	48
3.24	Different Concerns . . . . .	49
3.25	Positioning System Architecting . . . . .	50
3.26	What Can We Do to Improve the Environment? . . . . .	50
3.27	Conclusion . . . . .	52

4.1	Did you ever program a Video Recorder (VCR or PVR)? . . . . .	54
4.2	Product Creation Cycle . . . . .	55
4.3	2 Levels of Experience . . . . .	56
4.4	Bridging the gap between Experience and Engineering . . . . .	56
4.5	Example Time Shift recording . . . . .	57
4.6	What if? . . . . .	58
4.7	OOTI workshop 2001 . . . . .	59
4.8	Factors influencing the User Experience . . . . .	59
4.9	Infinite Experience Space . . . . .	60
4.10	Key Success Factor: Feedback . . . . .	61
4.11	The world of the construction . . . . .	62
4.12	Product Creation is much more than Engineering . . . . .	63
4.13	Educational Material per education stage . . . . .	64
4.14	Changing Education model in time . . . . .	64
4.15	Increasing Initiative required . . . . .	65
4.16	Architecting for Humans . . . . .	66
4.17	Experience Transfer . . . . .	66
5.1	What are the requirements for these products? . . . . .	70
5.2	User access point to long food-chain . . . . .	70
5.3	"Fuzzy expectations" and "SMART descriptions" . . . . .	71
5.4	Supply Chain Stakeholders . . . . .	72
5.5	Problem Statement . . . . .	73
5.6	Problem (2): From Imagination to Formalization . . . . .	74
5.7	Theory: Subcontractors require SMART relation . . . . .	74
5.8	Critical Success Factor: Mutual understanding . . . . .	75
5.9	Views on Aggregation; Why SMART is needed . . . . .	75
5.10	The "Fuzzy" needs of the User . . . . .	76
5.11	The "Fuzzy" needs of the Provider . . . . .	76
5.12	The "SMART" world of the Design . . . . .	77
5.13	Specifiable characteristics . . . . .	77
5.14	Response Time: Latency Budget . . . . .	78
5.15	Interaction or Irritation? . . . . .	78
5.16	Image Quality . . . . .	79
5.17	Fashionable . . . . .	79
5.18	From SMART to Fuzzy . . . . .	80
5.19	Complementing views . . . . .	81
6.1	Example product: mobile infotainment . . . . .	83
6.2	Value chain . . . . .	85
6.3	Stakeholders and concerns . . . . .	85
6.4	Internal stakeholders . . . . .	86

6.5	The "CAFCR" model . . . . .	87
6.6	Five viewpoints for an architecture. The task of the architect is to integrate all these viewpoints, in order to get a <i>valuable, usable</i> and <i>feasible</i> product. . . . .	88
6.7	The abstracted customer . . . . .	89
6.8	The quality needles are generic integrating concepts through the 5 CAFCR views . . . . .	89
6.9	Example security through all views . . . . .	90
6.10	Role of views . . . . .	91
6.11	Active listening: the art of the receiver to decode the message . . .	92
6.12	Intense interaction needed for mutual understanding . . . . .	92
6.13	Mutual understanding as function of time . . . . .	93
6.14	Story telling method . . . . .	94
6.15	How do these stakeholders communicate? . . . . .	95
6.16	Summary . . . . .	95
7.1	Did you ever program a VCR? . . . . .	97
7.2	Product Creation Cycle . . . . .	98
7.3	Bridging the gap between Human Experience and Engineering . .	99
7.4	Architecting visualized . . . . .	100
7.5	Five viewpoints for an architecture. The task of the architect is to integrate all these viewpoints, in order to get a <i>valuable, usable</i> and <i>feasible</i> product. . . . .	101
7.6	Guiding <b>how</b> by providing five <i>how</i> -viewpoints . . . . .	101
7.7	The architect integrating all specialist teamplayers . . . . .	102
7.8	Required architect know-how per view, typical for current architects and the preferred profile. . . . .	102
7.9	The architect maintains technical roots . . . . .	103
8.1	The product . . . . .	105
8.2	What is Architecting? . . . . .	106
8.3	Philips management objectives w.r.t. Campus . . . . .	106
8.4	The architects vision . . . . .	107
8.5	After user amendment . . . . .	108
8.6	Space impression . . . . .	109
8.7	The technical side of the architecture . . . . .	109
8.8	WDC architecting mapped on "CAFCR" . . . . .	110
8.9	Example product: Digital Video Recorder . . . . .	111
8.10	The technical side of the architecture . . . . .	112
8.11	Product architecting mapped on "CAFCR" . . . . .	112
8.12	Architecting dynamics . . . . .	113
8.13	Bridging 2 worlds . . . . .	114

# List of Tables

4.1	<i>Construction limits intrude in Experience</i>	57
4.2	<i>How to "SMART" en Experience?</i>	60
4.3	<i>Engineers are educated in construction disciplines</i>	62
4.4	<i>Prerequisites for continuous successfull product creation</i>	65
5.1	<i>The meaning of SMART</i>	68
8.1	<i>Objectives of the Philips management w.r.t. the Campus</i>	105
8.2	<i>Subgoals w.r.t. the Campus</i>	107
8.3	<i>Wishes and concerns of the inhabitants</i>	108
8.4	<i>Consumer Objectives</i>	111

# Introduction

This book bundles the articles about Human Measure and Architecting.

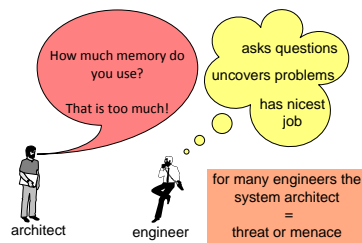
At this moment the book is in its early infancy. Most articles are updated based on feedback from readers and students. The most up to date version of the articles can always be found at [7]. The same information can be found here in presentation format.

Chapters can be read as autonomous units.



# Chapter 1

## The System Architect; Meddler or Hero?



### 1.1 Introduction

This lecture is presented at the DoVo<sup>1</sup> lecture series. This lecture series is an "institute" within Philips Research. Three short lectures (20 minutes) per session are used to share research work between all researchers. This long history also means that many traditions or rituals are followed, such as the opening which positions the speaker in the research line organization. The last section of this article "CBA" is also tradition.

The presentation itself focuses on a case, which is used to explain the contribution of the architect and the tension this causes in an organisation. Some background material is added about what an architect looks like and about the relation between architecture and research.

Figure 1.1 shows the position of the speaker in the organisation. However in this, somewhat satiric, diagram other (more) important organisational dimensions are shown:

- program and projects, where a program is result driven

---

<sup>1</sup> Donderdag Ochtend Voordrachten: Thursday morning presentation.

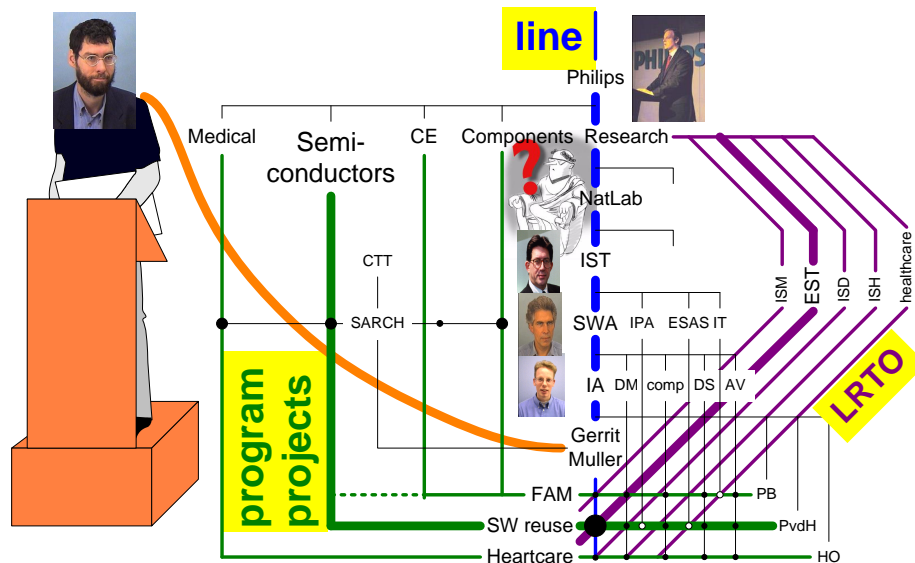


Figure 1.1: The position within the organisation

- LRTO (Long Range Technical Objectives), research wide objective driven management
- know how transfer, via the CTT (Centre for Technical Training)

Another interesting tradition is that groups are more often identified by the name of the leader than by the technical competence. This emphasis on the human (leader) contribution is nice.

Recommended literature is the book by Rechtin, "The Art of Systems Architecting" [14].

## 1.2 Practical experience: memory usage in a medical workstation.

The medical workstation[8] is an add-on product to existing X-ray systems, introduced in the market in 1992. X-ray systems used to print the imaging results directly on film, by means of a so called CRT-copy, an exact copy of the monitor display on film. The workstation is positioned between X-ray system and printer and adds formatting and layout capabilities. One workstation can serve multiple examination rooms, see figure 1.2.

The software designers of this product did an excellent job, applying many new technologies in a fruitful way. However many integral design aspects did not get the right attention level, for example memory usage.

Figure 1.3 shows the memory usage at the beginning of the integration phase.

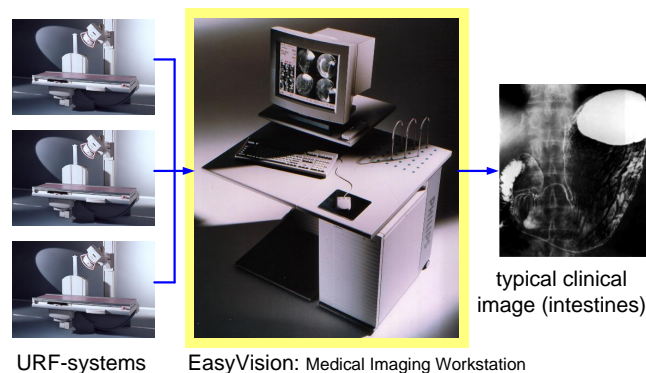


Figure 1.2: Practical experience; a medical workstation

The figure clearly shows a disastrous problem: the system needs much more memory (ca 200 MByte) than available as physical memory (64 MByte). At the bottom of the figure the performance of the system is shown as a function of the memory use. A minor shortage of memory is handled by the virtual memory system, but a major shortage leads to an unacceptable drop in performance.

The architect starts to ask questions about the memory usage, measures it, makes models and budgets. The result is that in cooperation with the software engineers an iterative redesign was implemented. This redesign realized an acceptable memory usage, see figure 1.4.

Models and budgets are important means of the architect. Figure 1.5 shows the memory budget, which is based on a process decomposition of the system. This process decomposition enables a manageable granularity of the budget and sufficient implementation freedom for designers. It also enables measurement and verification, because the operating system and the analysis tools use process boundaries as natural resource management boundaries.

The SW engineers of the medical workstation did not experience any performance problem while creating their components, because every individual component fits easily in the available memory. Only when the system is integrated and used under production conditions the performance problem becomes visible. This late visibility and detection of problems is quite normal in the development of complex systems.

Projects run without (visible) problems during the decomposition phases. All components builders are happily designing, making and testing their component. When the integration begins problems become visible. Figure 1.6 visualizes this process. The invisible problems cause a significant delay<sup>2</sup>.

<sup>2</sup>This is also known as the *95% ready syndrome*, when the project members declare to at 95%, then actually more than half of the work still needs to be done.

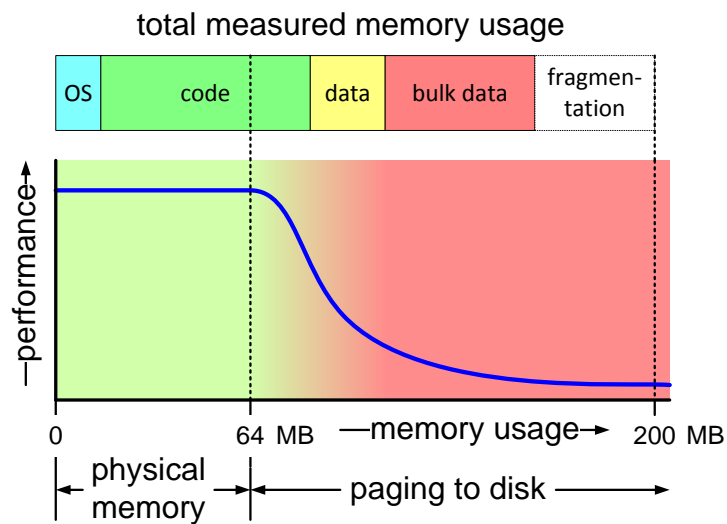


Figure 1.3: Problem: unlimited memory consumption

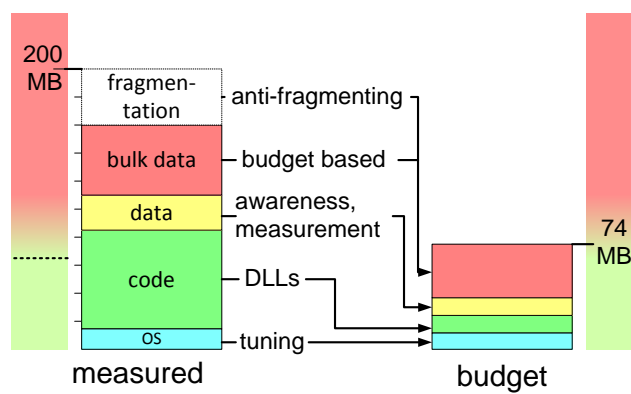


Figure 1.4: Solution: measure and redesign

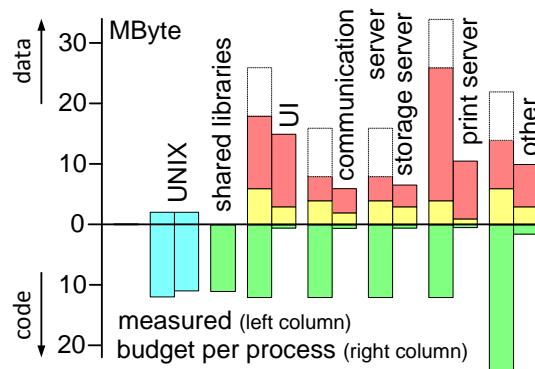


Figure 1.5: Method: process based budgeting

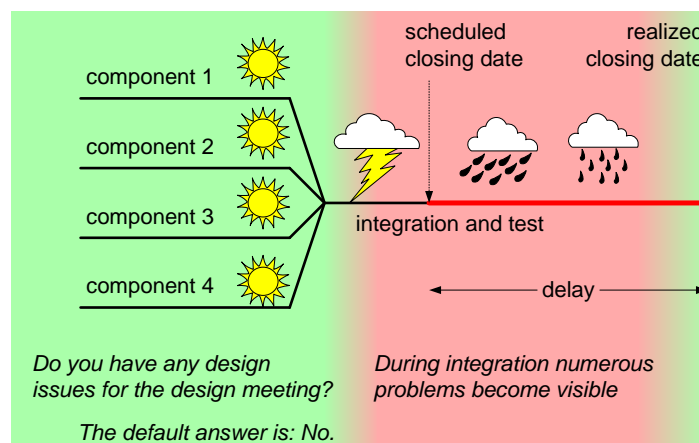


Figure 1.6: Integration uncovers hidden problems

## 1.3 Introducing an Architect

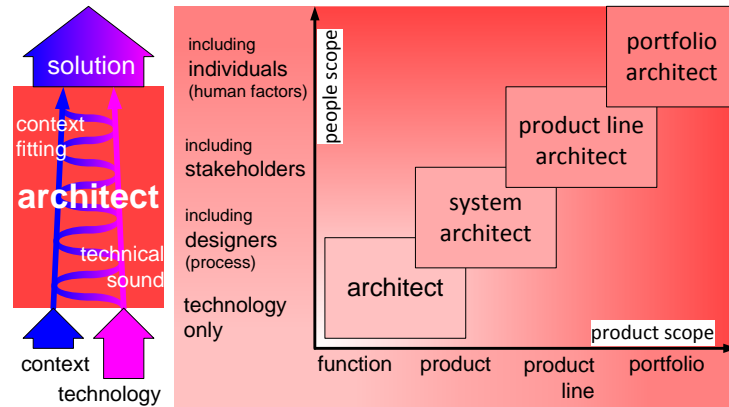


Figure 1.7: Architecting scope

One of the main causes of the late visibility of problems is the limited context awareness of most component engineers. Most of these engineers are simply not aware of potential problems! This lack of awareness is reflected in the difficulty to plan design meetings. Quite often the engineers don't see the benefit of such a meeting, because they don't see any issues to be discussed.

Many organisations don't have explicit system architects. Sometimes the *best in class* technical specialist gets the architect title imposed. In both cases the introduction of a broad system architect causes a shock effect in the organisation.

Figure 3.4 shows that the scope of architects widely varies. The common denominator for all these architects is the bridge function between context and technology (or problem and solution). An architect needs sufficient know-how to understand the context as well as the technology, in order to design a solution, which fits in the context and is technical sound at the same time.

In general increasing the product scope of an architect coincides with an increase in people scope at the same time.

Figure 1.8 shows the phases an organisation is going through in a typical project where an architect is introduced.

As long as individual designers can work independently the collective mood is great, while an architect is mostly perceived as a threat or a menace, see also figure 1.9. As soon as the integration starts all invisible problems suddenly become visible, the beginning of a crisis, which changes the mood to poor.

An architect who proves himself in this difficult stage, by hard work, brainstorming, trouble shooting and problem solving earns a lot of credit. This changes the appreciation of the architect dramatically, suddenly he becomes an indispensable team member! After completion of the integration the mood returns to good. In a

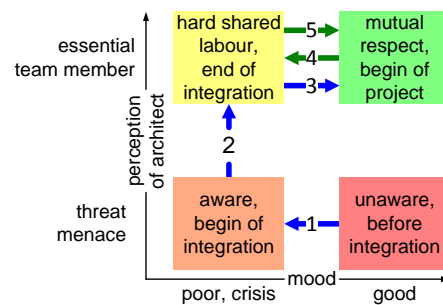


Figure 1.8: Architecture awareness evolution

next project, again during integration the mood will degrade, even excellent architects will not prevent this. However this crisis will be less severe.

An architect always starts to ask questions, to build up understanding and overview. While sampling the problem and solution domain in this way, he always discovers some weak spots. The identification of problems and risks is often based on a judgement or an opinion.

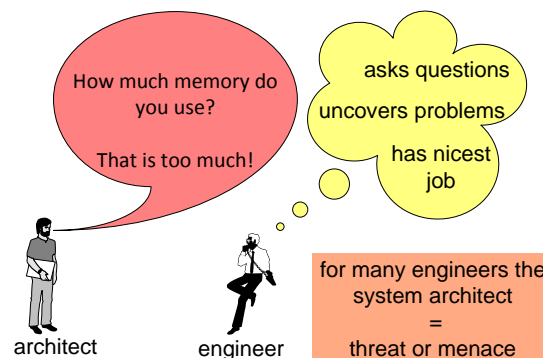


Figure 1.9: The engineer's perception of the architect

The judgement or the opinion is based on a sample of all data, fitting in the limited available time. Despite the incompleteness of the data and despite a lack of domain and solution know-how the architect forms an opinion anyway. The architect does the top level design, the nice hand-waving work, without being too concerned with the nasty details.

Whenever you want to benefit from the architect's expertise, by asking for a solution, the architect only worsens the problem, by showing even more hidden problems.

## 1.4 What is architecting?

Architecting in product creation spans from *understanding* the **why**, via *describing* the **what** to *guiding* the **how**, as shown in figure 7.4. Or in even more popular terms: *do the right things* and *do the things right*

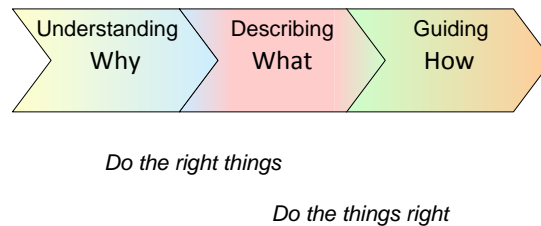


Figure 1.10: Architecting visualized

Architecting is a job which is done by all members of the product creation team, however the architect is responsible for the consistency and balance of **why**, **what** and **how**

A useful top level decomposition of an architecture is provided by the so-called "CAFCR" model, as shown in figure 7.5. 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!).

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).

Figure 1.12 shows these 5 views with some relevant issues with respect to the illustrated memory usage. The customer objectives are expressed in a number of keydrivers, constrained by a street price of 50k\$.

The application and functional view are here shown together, expressed by a typical case with 3 connected X-ray system, where an examination has a typical size of 20 images, which are auto-printed on 3 film sheets.

The conceptual view contains a decomposition, amongst others in import, database and print servers. It also contains concepts to constrain the memory usage, such as anti-fragmentation and dynamic link libraries.

The realization view contains the actual measured memory usage as well as the budgeted usage.

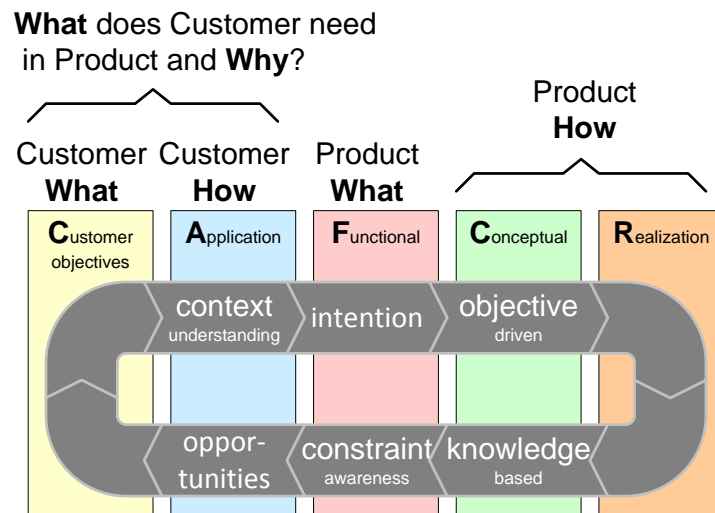


Figure 1.11: Five viewpoints for an architecture. The task of the architect is to integrate all these viewpoints, in order to get a *valuable, usable and feasible* product.

The **how** of the product is created by many specialists. The **how** is guided by the architecture. At least 5 views are required for guidance:

- functional decomposition
- construction decomposition
- allocation of functions to construction elements
- infrastructure
- integrating concepts

Figure 7.6 visualizes these 5 **how** views.

Figure 1.14 and 1.15 shows a question generators, which can be used to uncover potential problems. This question generator is based on a discrete 3-dimensional space, where every point in this space can be used to formulate a question. The axis of this space are:

- functions
- (HW) components
- characteristics

The question in any point in space looks like:

”How about *characteristic c* in *HW component h* when performing *function f*?”

Nearly all questions formulated in this way will get an answer *unknown*, which in most cases means *here is a potential problem*.

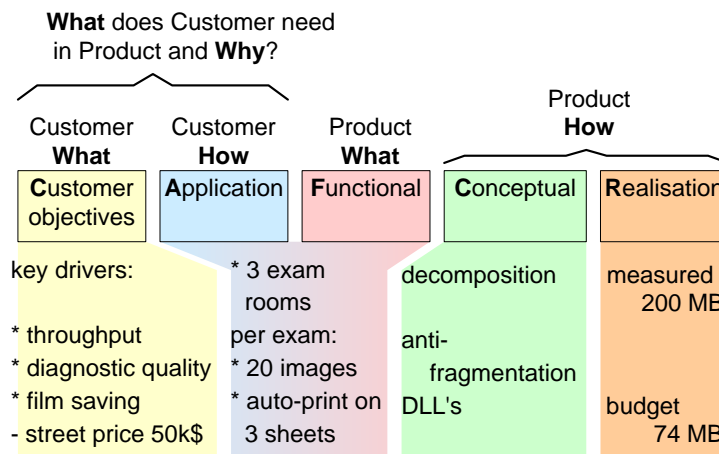


Figure 1.12: The customer issues which are relevant for the case illustrated in this article

Note that a good architect uses a more refined question generator, which uses a priori know-how to select the really relevant questions. The a priori know-how includes:

- importance, value of functions and performance
- how critical, sensitive technical solutions are
- people and organisation bias, strengths and weaknesses

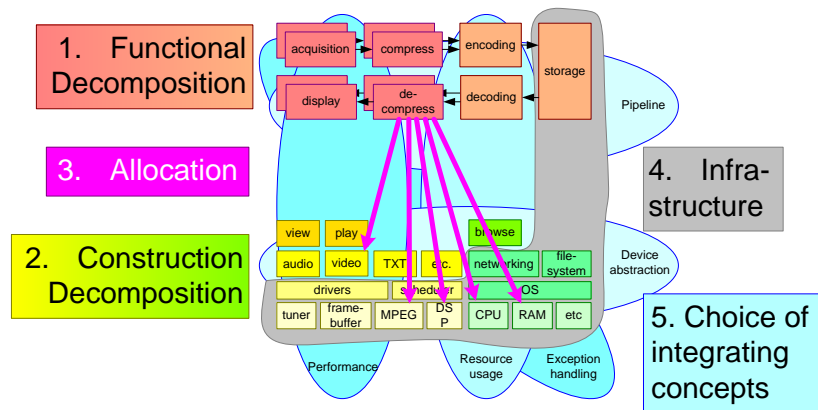


Figure 1.13: Guiding **how** by providing five *how*-viewpoints

How about the **<characteristic>**  
of the **<component>**  
when performing **<function>**?

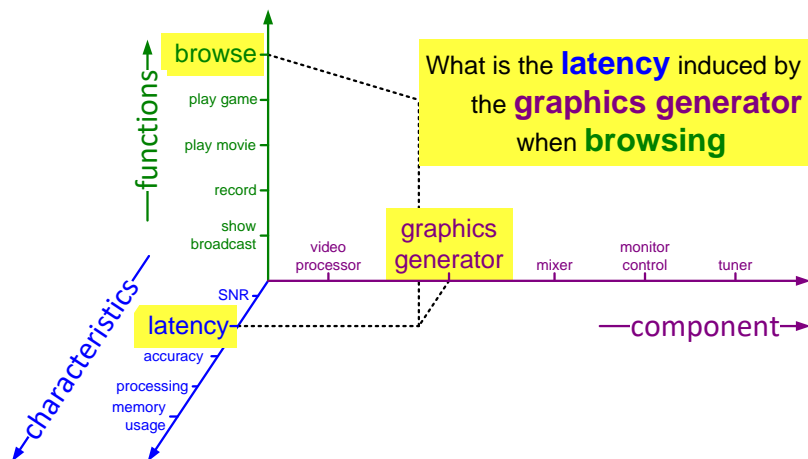


Figure 1.14: Question generator

How about the **<characteristic>**  
of the **<component>**  
when performing **<function>**?

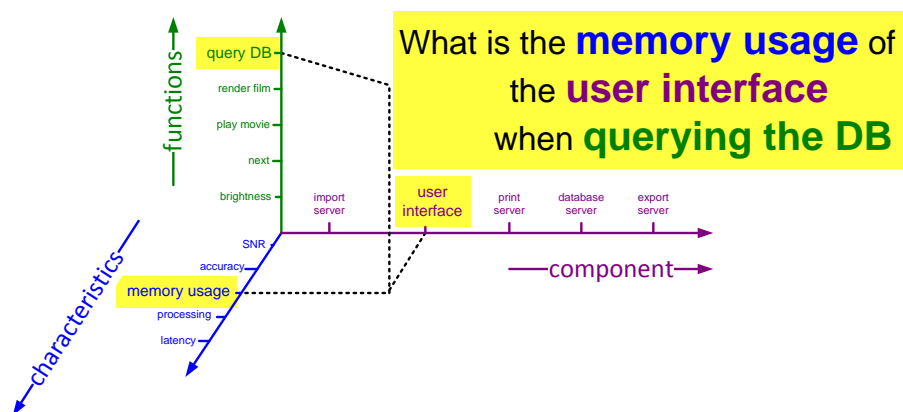


Figure 1.15: Question generator

## 1.5 CBA: Conclusions, Benchmarking and Acknowledgements

### 1.5.1 Conclusions

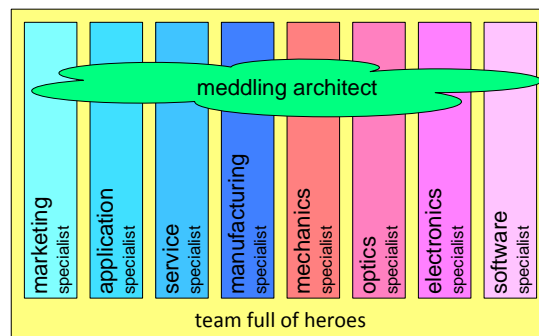


Figure 1.16: The meddling architect as complementing factor in a team full of heroes

The work of the architect is always overlapping with the work of others, see figure 7.7. The integration of views is the main added value of the architect. Sometimes the architect is meddling in the work area of specialists, with the intention to serve the overall objective. The architect can only be successful by virtue of the rest of the team, so the architect is not the hero, all team members are heroes.

### 1.5.2 Benchmarking

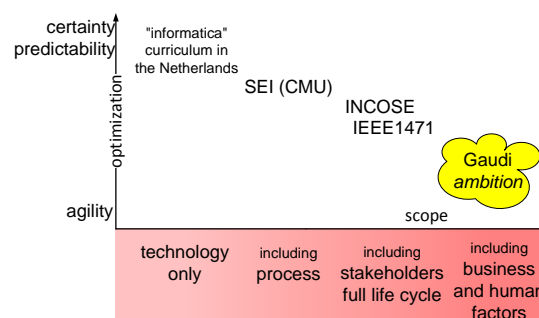


Figure 1.17: Positioning the Gaudí research ambition in the worldwide efforts

Most software engineering oriented institutes in the world, such as SEI(CMU) stress the importance of systematic approaches and the use of processes. This

emphasis often results in formalization, which endangers the flexibility and adaptivity to fast changing technology and markets<sup>3</sup>

For wider scopes formalization is more difficult and less fruitful. Insight, understanding and overview are more important in a broad perspective. Figure 2.8 positions several worldwide activities with respect to formalization level and scope.

The System Engineering community, which is mostly flourishing in the military and aerospace industry, is very mature with respect to requirements engineering, stakeholders and life cycle management. Part of that work is consolidated in "standards" for best practices, such as IEEE1471.

The ambition of the Gaudí project is to provide more insight in the *art* part of architecting, which quite often is related to human aspects.

### 1.5.3 Acknowledgements

Figure 1.18 shows some of the participants of the SARCH courses. The participants of these courses provide me always with valuable feedback and often trigger new insights.

Hammer, Dieter	Gijsbers, Rob	Penners, Maurice	van Gogh, Clemy	van der Sterren, William
Hoogenstraaten, Wil	Huis in 't Veld, Robert	America, Pierre	Wissink, Getty	Soede, Michiel
Mueller, Juergen	Joosten, Jan	Jaspers, Peter	Engelsma, Erwin	van Bommel, Luc
Gieles, Hans	Mulder, Alwin	Versteijlen, Joost	Stut, Wim	Krikhaar, Rene
Eggenhuisen, Huib	de Wit, Paul	Beelen, Peter	Luttikhuisen, Paul	van den Brink, Johan
Kloprogge, Raymond	Poesse, Jan	Blijd, Jarl	Bruin, Jan	Ham, Kees
Engel, Bas	Spaak, Wim	Dijkema, Marcel	Gooren, Huub	Bos, Erik
van Rijnsoever, Bart	Thus, Frank	Roelandt, Werner	den Dekker, Wim	Pijpers, Frank
Driesen, JGH	van Velden, Jeroen	Janson, Paul	van der Laak, Eric	Medema, Jeroen
Schelkers, Raymond	van Venrooy, Roland	Bandakka, Mahesh	Crins, Wim	Kaag, Bjorn
van der Heijden, Jaap	Dobbelsteen, Jan	Ledeboer, Jodie	Heerink, Lex	Giesselman, Timo
Vermeulen, Gerry	de Waal, Klaas	Geron, Nic	Schippers, Alef	Vos, Frans
Stroucken, Marc	Muijen, M	Zieringer, Peter	Schreppers, Jurgen	de Greef, Pierre
Wijnstra, Jan Gerben	Peters, Jo	Beuk, Leo	Deckers, Robert	Fischer, Stefan
Algra, Egbert	van Bommel, Pieter Jan	Koolen, Gertjan	van Balen, Auke	Pu, Xuemei
Derks, Frans	Thijssen, Henk	Koushik, Sudeendra	Huiban, Cristian	Boom, Sjirk
Faber, Albert	Boot, Gert Jan	Milosevski, Vlatko	van Loon, Gerard	ten Pierick, Henk
Aarts, Peter	Vullings, Erik	van den Broek, Ger	van den Heuvel, Patrick	Stroucken, Louis
Watabe, Yasuma	Vermeer, Ad	de Kruij, Peter	Lobo, Lincoln	Young Tai Liu
van Ouwerkerk, H	Peeters, Bob	Daenen, Steven	van de Meulenhof, Dennis	van der Steen, Marcel
Huijnen, Ton	Obbink, Henk	Soepenbergh, Gerben	Houteppen, Rob	Siereveld, Ad
Gonot, Mathieu	Bas, Han	Bingley, Peter	Hofsink, Robert	van Bakel, Gerian
van Splunter, Andre	Rankers, Adrian	Follon, Roel	Buurman, Hans	Engbers, Rene
van Rooijen, Joost	Akiwumi-Assani, Olu	Elzinga, Onno	Zondag, Eddy	van Wetten, Frank
Verberkt, Mark	Gopalan, Rajaraman	van den Donker, Piet	Veldmans, Ferdinand	Stevens, Frank
van der Linden, Wim	Misdorn, Han	Zwaans, Ben	Merkus, Paul	Wubben, Rob
Patrzaiek, Jarek	Schatorie, John	Harmsze, Françoise	van Tuijl, Frank	Schellingerhout, Nico
Vergoossen, Theo	Boer, Richard	Jansen, Tom	Wouters, Kees	Vugts, John

Figure 1.18: A small subset of contributors, here mostly participants of the System Architecture course

Jürgen Müller again critically reviewed the presentation, helped to streamline it and discussed the right sequence of presenting. William van der Sterren and Peter

<sup>3</sup>Strictly speaking formalization and agility are not contradictions. Agile formalization can be supportive. The actual danger is in the less skilled people applying the systematic approaches in rigid ways.

van den Hamer suggested a lot of improvements. Jaap van der Heijden pointed out the areas of interest for the target audience.

Pierre America, Jaap van der Heijden, Niek Lambert and Milan van den Muyzenberg patiently listened to the trial run and politely pointed out the missing steps and unclear issues.

#### 1.5.4 Gaudí homepage

[http://nlwww.natlab.research.philips.com:8080/  
research/swa\\_group/muller/](http://nlwww.natlab.research.philips.com:8080/research/swa_group/muller/)

*containing:*

more than 30 recent articles and or presentations  
course material SARCH, ESA stakeholders, OOTI req eng

*links on this homepage:*

this presentation	<a href="#">MeddlerOrSaviorSlides.pdf</a>
annotated text	<a href="#">MeddlerOrSaviorPaper.pdf</a>
background information Medical Imaging	<a href="#">MedicalImagingPaper.pdf</a>
original documentation Medical Imaging	<a href="#">oldPresentations.html</a>

Figure 1.19: Gaudí homepage

Figure 1.19 shows the URL for the Gaudí homepage, see[7] for the www internet URL.

See also the bibliography for more recommended reading.

## 1.6 Does architecture belong in the research laboratory?

Research laboratories are used to in-depth technology research, where system know how is mostly required to build a proof of concept. Architecting at the other hand is very broad and not tangible. Figure 1.20 shows a model for technology management[3], based upon 3 phase cycle.

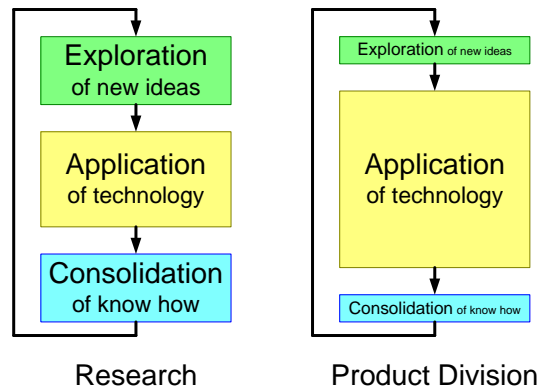


Figure 1.20: Architecture and research? The technology management cycle

One of the main functions of research is technology exploration, where the technology application is needed for learning and proof of concept. The consolidation is needed for the transfer.

Product divisions focus on short and medium term business objectives, research is taking care of the long term. Redefinition of architecting as enabling technology to specify, design and integrate complex systems, fits architecting in a natural way in this technology management model. It makes quite a lot of sense to explore architecting methods as well as consolidate architecting methods.

However the application of architecting methods often needs the full product creation context, which means that the application often happens in close cooperation with the industry, the so called **industry as laboratory**.

The next question is: *Is architecting research scientific?*. The answer of this question requires a philosophical basis, what do we mean with *scientific*? Figure 1.21 shows several sciences as a spectrum with respect to the hardness level of the scientific methods. This figure shows that architecting methods span a significant range of scientific methods. Or in other words the architecture researcher borrows methods from other scientific paradigms, striving for as hard results as possible<sup>4</sup>

<sup>4</sup>Which means that the results can be very soft. How to distinguish plausible results from an integer researcher from convincing results from a charlatan?

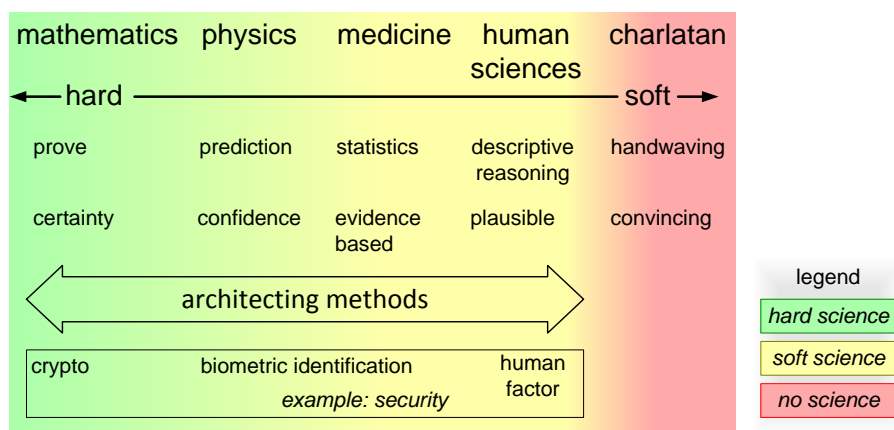


Figure 1.21: Is architecting scientific?

## 1.7 What is an architect?

The architect is good technical educated engineer, who has grown into an architect, see [6]. The main growth direction are:

- technical generalist
- business insight
- process insight
- human insight

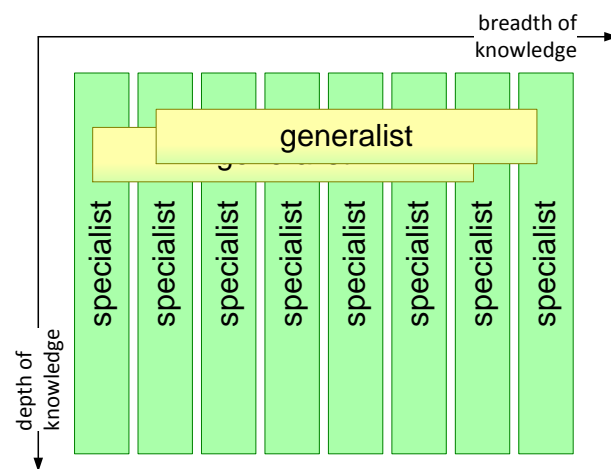


Figure 1.22: The architect as integrator

The role of the architect is an integrator role, as shown in figure 1.22, which is highly complementary to the specialists.

The real world is less black and white, a complete spectrum exists between specialists and generalists, as shown in figure 7.9. Architects must have sufficient roots in the technical domain, which means that they must experience engineering in at least one discipline. Later when growing in the above mentioned directions the difficult challenge is to maintain sufficient technical know how and feeling.

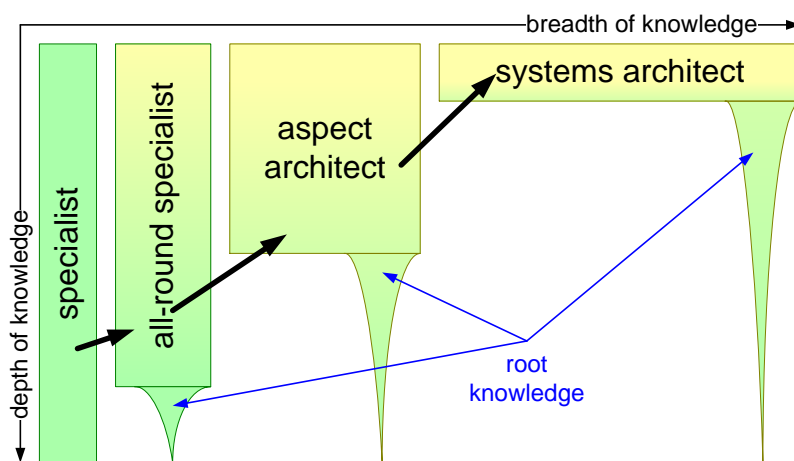
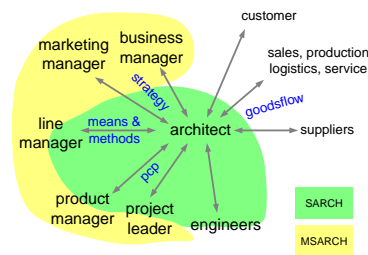


Figure 1.23: The architect maintains technical roots

## Chapter 2

# The Importance of System Architecting for Development



### 2.1 Introduction

Systems architecting is one of the integrating disciplines in Product Creation. Complementary integral teamplayers, such as project leaders, architects and product managers can form the core of highly effective product creation teams.

Several efforts are ongoing to help potential architects in developing the needed skills and obtaining sufficient knowhow, such as the architecture school at research, the ESA course and the SARCH course.

The next step in enabling these complementary core teams, is to address the managerial context of the system architect. A special course, in the form of a two day workshop is developed to create a shared insight in the role of the architect and architecting.

## 2.2 The Challenge

The functionality and performance of products is ever increasing. The increase of functionality and performance results in an increase of the effort to make new products. Historical data shows that the effort increase is also exponential, like Moore's law for electronics. The increase of designer productivity (new functionality or performance increase per designer year) is rather limited, despite many promising reuse, platform, COTS (Commercial Off The Shelf) et cetera approaches.

Figure 2.1 shows the challenge we are facing: to increase the designer productivity dramatically, in order to keep the product creation teams at a manageable size.

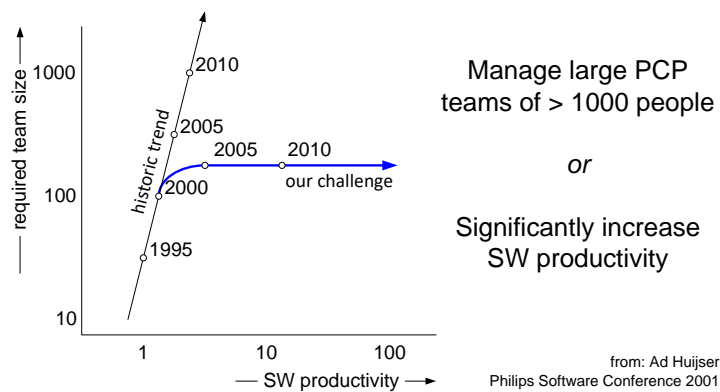


Figure 2.1: The Challenge

In the keynote speech at the Philips Software Conference 2001 Ad Huijser showed that this challenge must be addressed by a multitude of measures, ranging from business to people management; see figure 2.2. System architecture is one of many measures to attack the challenge of dramatically increasing the designer productivity.

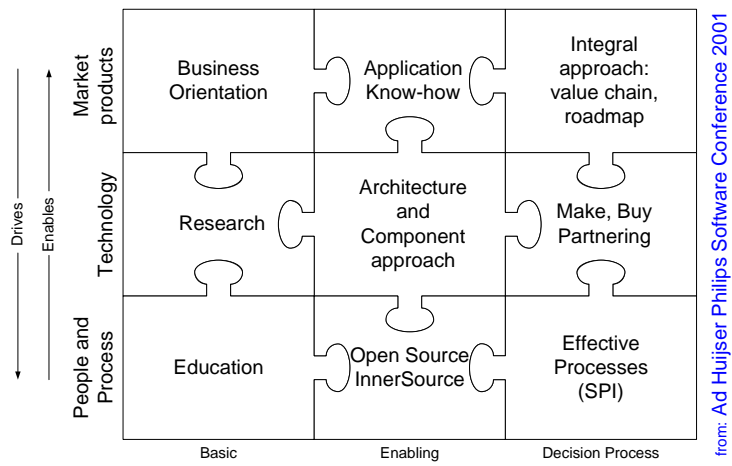


Figure 2.2: When all pieces fit ...

## 2.3 Architecting

The architecting courses at this moment use a few fundamental concepts as the basis of system architecting. A simplified process decomposition, see figure 2.3, is used to explain the context of system architecting in the business context.

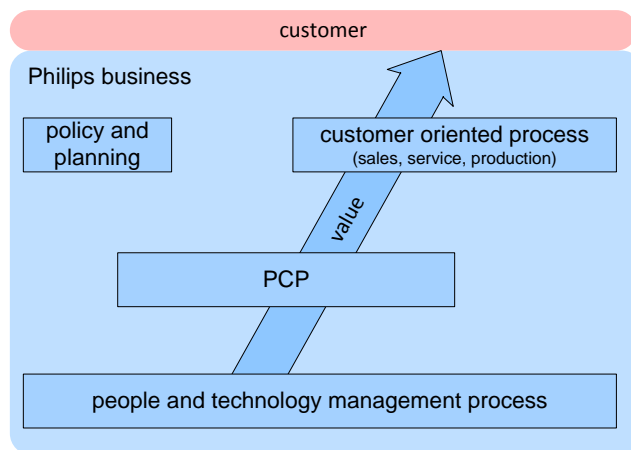


Figure 2.3: Simplified process view

Figure 2.4 shows the system architecting process overlayed on this process decomposition, as well as the relations with the other processes. Normally an architect will spend 80% of his time in product creation and 20% in product policy and planning.

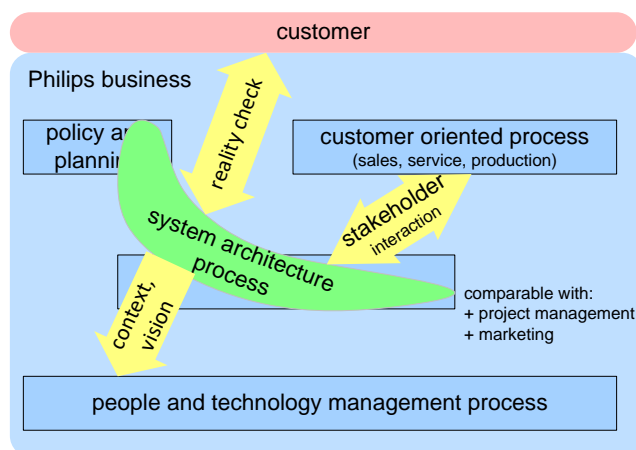


Figure 2.4: System architecture process

One of the fundamental messages is that architects must combine know how of

the solution (technology) with understanding of the problem (customer/application). The architect must play an independent role in considering all stakeholders interests and searching for an effective solution. The fundamental architecting activities are depicted in figure 7.4.

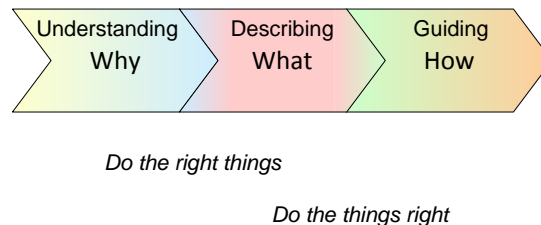


Figure 2.5: What is architecting?

During the course often the "CAFCR" model is used as simple reference model, see figure 6.5. This model is a refinement of figure 7.4

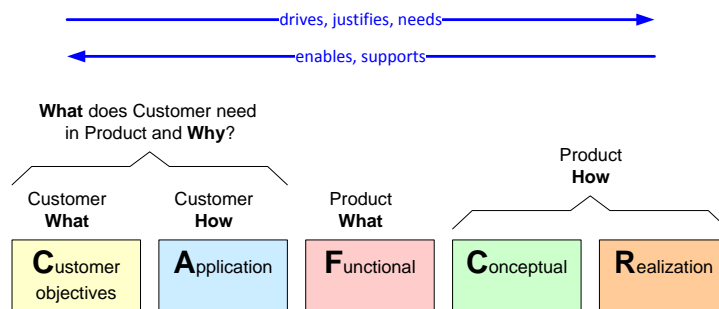


Figure 2.6: "CAFCR" model

Creating the solution is a collective effort of many designers and engineers. The architect is mostly guiding the implementation, the actual work is done by the designers and engineers. Guiding the implementation is done by providing guidelines and high level designs for many different viewpoints. Figure 7.6 shows some of the frequently occurring viewpoints for guiding the implementation. Note that many people think that the major task of the architect is to define the decomposition and to define and manage the interfaces of this decomposition. Figure 7.6 shows that architecting involves many more aspects and especially the integrating concepts are crucial to get working products.

Many more initiatives are ongoing in the world. Figure 2.8 defines the relative position of these initiatives and the ambition of the Gaudí project. The figure shows that the system architecting efforts are not heading for a fixed set of rigid procedures, but instead are stimulating architects to fill their personal toolbox with many

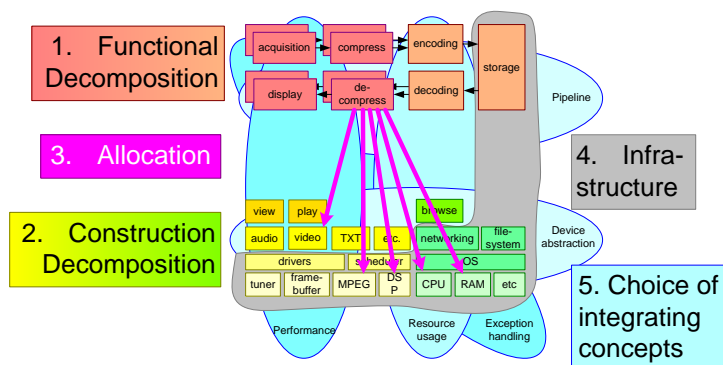


Figure 2.7: Guiding how

flexible tools to remain agile (responsive, adaptable, et cetera).

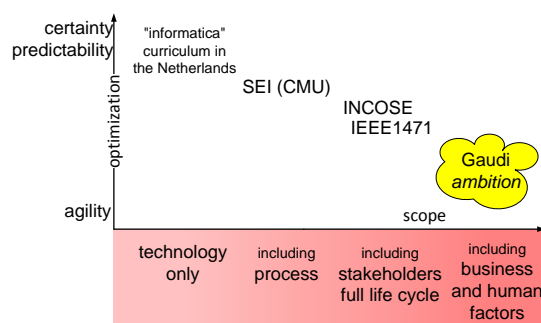


Figure 2.8: Gaudí ambition

The architect is the technical oriented integrating core team player. In most cases the project leader is operational organization oriented, and the product manager commercial oriented. These 3 roles exist recursively at other levels, from component level to product portfolio. Figure 2.9 shows this operational oriented hierarchy in product creation.

Figure 2.9 makes clear that the architecting role is present at multiple scopes. Parallel with the increase of the scope in the product direction the scope of the architect is increasing in the people direction, as shown in figure 3.4

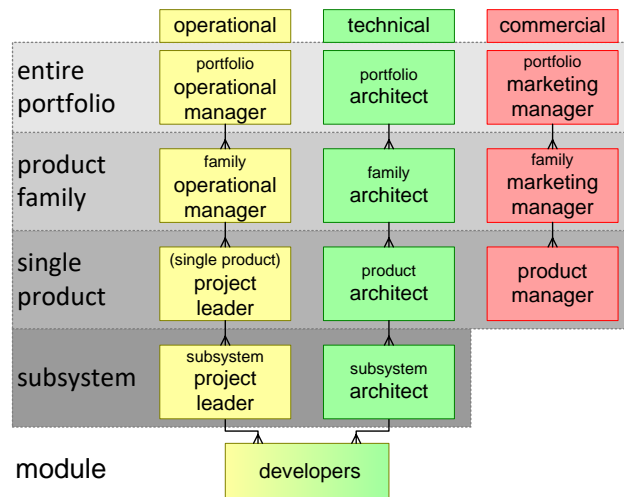


Figure 2.9: Operational hierarchy

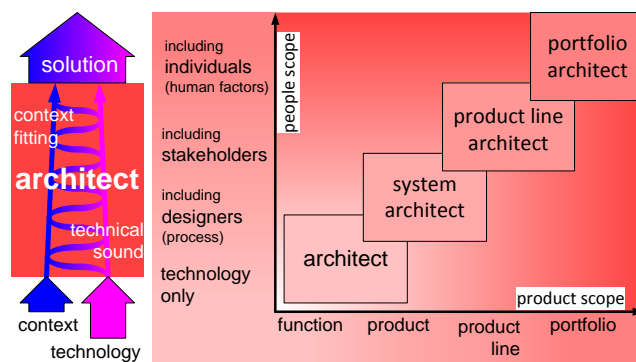


Figure 2.10: Architecting Scope

## 2.4 Architecting Courses

The two main types of architecting courses are:

- courses for (potential) architects and close relatives
- courses for the managerial context

Figure 2.11 shows the architect and his stakeholders, and positions the SARCH and the MSARCH in this landscape.

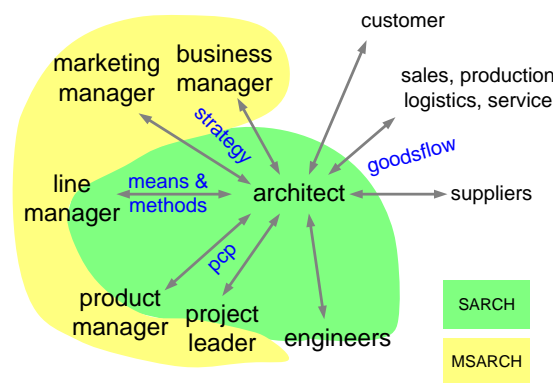


Figure 2.11: Stakeholders Architect

The course tries to stimulate the (potential) architect to broaden his profile, as shown in figure 7.8. Note that this broadened scope will somewhat reduce his technology involvement. The architect has to learn to cooperate with the designers and engineers to compensate for this reduction. For most architects this is a difficult step, managers should coach and help architects through this transformation.

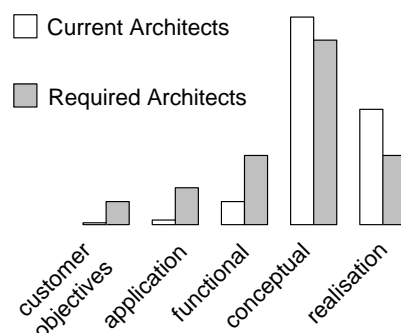


Figure 2.12: Profile of System Architect

A first attempt to define a curriculum for architects is shown in figure 2.13. At the top of the figure the growth path of a system architect[6] is shown. Below the courses or course subjects are shown which fit in the architect career path. Note that this is not a unified list for all architects. Instead it is a palet of courses, where the architect must select the courses which best fit his current needs. In color coding is indicated if courses are available inetrnal or external.

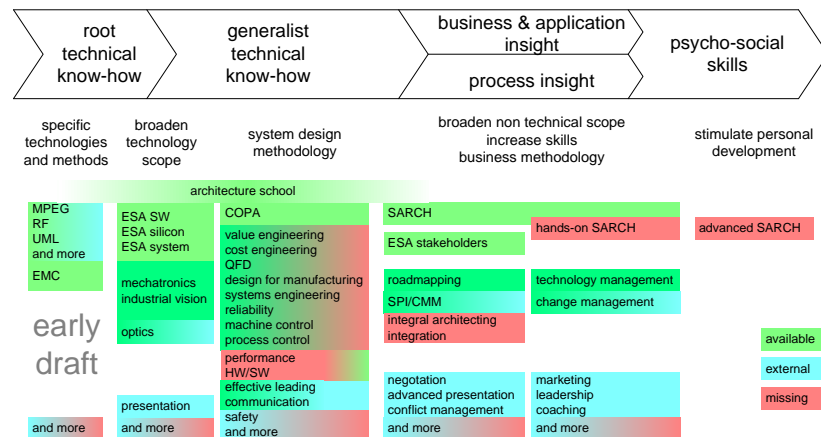


Figure 2.13: Draft System Architect Curriculum

Figure 2.14 shows the current status of courses within Philips. The first SARCH course has been given end of 1999, the ESA course started mid 2000 and the MSARCH was first given begin 2002.

Course	Abbreviation	number of courses upto May 2002	appr. total participants	Lecturers
System Architecture	SARCH	15	230	Gerrit Muller 2002 H2: others
Embedded Systems Architecting; Stakeholders	ESA	5	80	Pierre America Frank Pijpers
System Architecting for Managers	MSARCH	1	12	Gerrit Muller

Figure 2.14: Course status in Philips

Figure 2.15 shows the goal of the 2 day management SARCH workshop. Many architects complain that their (managerial) context does not share the same view on

architecting, which limit them in performing the job in the broad role as described here. One of the main messages to the architects is that they themselves must earn credibility and create visibility. Philips may gain a lot of effectiveness if the managers are working and facilitating in the same direction, by sharing the same vision on architecting and architects.

managerial awareness of:

- + what is architecting
- + business impact of architecting
- + role and profile of an architect

to

- + enable integral approach
- + stimulate architects to substantially contribute:
  - \* at business level
  - \* to strategic goals
  - \* from technological strength

Figure 2.15: Goals of 2-day Management SARCH course

The challenge for a Management SARCH is to balance the theory (200+ sheets of SARCH material), with practical illustration and even more important active hands-on work. It is extremely dangerous if the management course only consists of glossy, well defined models and processes. That creates an illusion of understanding, while the subtle relationships, conflicting interests and non ideal behavior are not experienced. That is the main reason that more than half of the time is spent (inter)active. Altogether a very loaded program is followed, see figure 2.16.

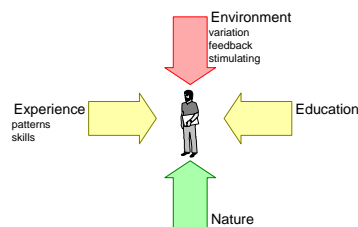
session	subject
day 1 morning	positioning the System Architecture Process Product Creation Process  product families, generic developments
day 1 afternoon	role and task of the system architect profile of the system architect  documentation, reviewing and other supportive processes
day 2 morning	requirements capturing, roadmapping
day 2 afternoon	HRM aspects; selection, appraisal, career path, etcetera wrap up, expectations, how to continue, evaluation

Figure 2.16: Program of 2-day Management SARCH

When approaching management teams the investment of 2 full days is often a hurdle. Nevertheless to reach the desired objectives this is the minimum time needed. If managers are not yet ripe to show commitment by investing these two days, than other bootstrapping activities, like given a short interactive presentation, are needed before forcing this course on them. Only self motivated teams will benefit from such a course.

## Chapter 3

# Decomposing the Architect; What are Critical Success Factors?



### 3.1 Introduction

One of the big challenges of today is: How do we get more, and effective, system architects? At Philips and the Embedded Systems Institute we have been very successful in teaching the non-technical aspects of systems architecting to people. This course, called SARCH, has been given 36 times (May 2006) to about 570 participants. We also provide the Embedded Systems Architecting course (ESA), that has been given more than 20 times to more than 300 participants, which addresses the technical broadening of designers. We identified a number of missing steps in between these courses: addressing multi-disciplinary design. We fill this hole by "single aspect" courses that address one aspect at the time, for instance, performance or reliability. The performance oriented course, that has been given 7 times to about 100 people, is also successful. The next course that we developed to fill this hole is the Multi-Objective System Architecting and Design (MOSAD) course. The evaluation after 3 courses revealed a problem: the participants are satisfied, but the teacher is not satisfied. The dissatisfaction of the teacher is that the participants pick up many submethods and techniques provided in the course, but they struggle to integrate this into an architecting approach.

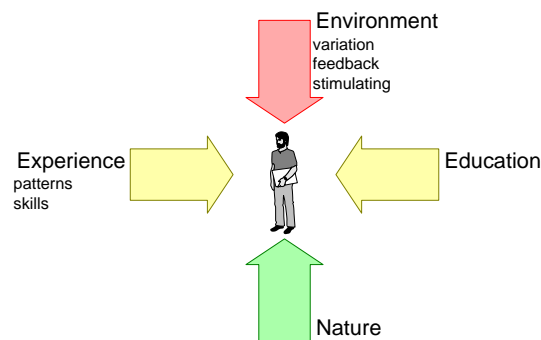


Figure 3.1: Decomposing Contributing Factors

This conclusion triggered the analysis of critical success factors for system architects. We decomposed these factors into four categories: *education*, *experience*, *environment*, and *nature*, as shown in Figure 3.1. We will discuss these four categories in separate sections. We will start with a section about the architect, to create a baseline for the further analysis.

## 3.2 What is an Architect?

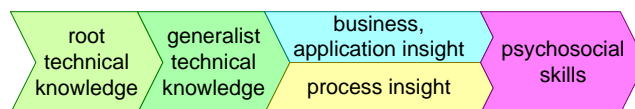


Figure 3.2: Typical Development of a System Architect

System architects need a wide range of knowledge, skills and experience to be effective. Figure 3.2 shows a typical development of a system architect.

The system architect is rooted in technology. A thorough understanding of a single technological subject is an essential underpinning. The next step is a broadening of the technical scope.

When the awakening system architect has reached a technological breadth, it will become obvious that most problems have a root cause outside of technology. Two main parallel streams are opened:

- The business side: the market, customers, value, competition, logistics, service aspects
- The process side: who is doing what and why

During this phase the system architect will broaden in these two dimensions. The system architect will view these dimensions from a technological perspective. Again when a sufficient level of understanding is attained an awareness starts to grow that people behave much less rationally than technical designs. The growing awareness of the psychological and the sociological aspects is the next phase of growth.

Most developers of complex high tech products are specialists. They need an in-depth understanding of the applicable technology to effectively guide the product development. The decomposition of the development work is most often optimized to create a work breakdown enabling these specialists to do their work with as much autonomy as possible.

Most generalists are constrained in the depth of their knowledge by normal human limitations, such as the amount of available time and the finite capacity of the human mind. The figure also shows that a generalist has somewhere his roots in in detailed technical knowledge. This root is important for the generalist himself, since it provides him with an anchor and a frame of reference. It is vital in the communication with other specialists, because it gives the generalist credibility.

Both generalists and specialists are needed. Specialists are needed for their in depth knowledge, while the generalists are needed for their general integrating ability. Normally there are much more specialists required than generalists. There are more functions in the Product Creation Process which benefit from a generalist profile. For instance the function of project-leader or tester both require a broad area of know how.

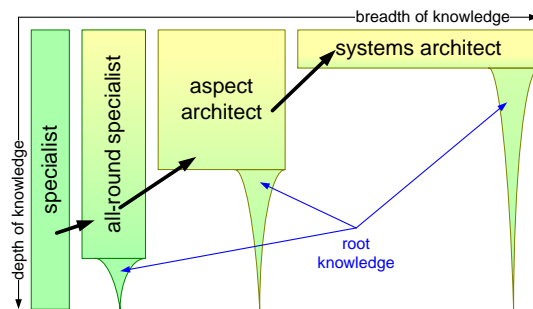


Figure 3.3: Growth in technical breadth, intermediate functions from specialist to system architect

Architects require a generalist profile, since one of their primary functions is to generate the top-level specification and design of the system. The step from a specialist to a generalist is of course not a binary transition. Figure 7.9 shows a more gradual spectrum from specialist to system architect. The arrows show that intermediate functions exist in larger product developments, which are natural

stepping stones for the awakening architect.

Examples of aspect architects are:

- subsystem architects
- SW, mechanics or electronics architects

For instance a software architect needs a significant in-depth knowledge of software engineering and technologies, in order to design the software architecture of the entire system. On the other hand a subsystem architect requires multi-disciplinary knowledge, however the limited scope reduces the required breadth to a hopefully realistic level.

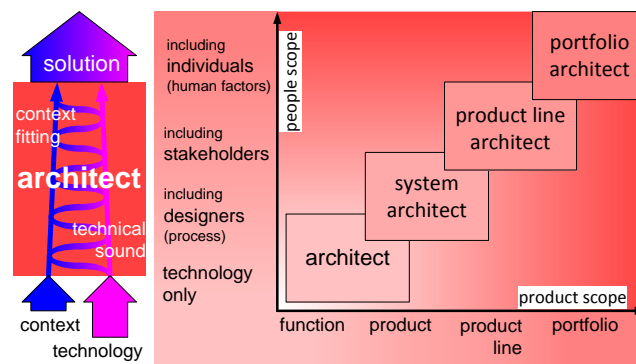


Figure 3.4: Different Architecting Scopes

Many products are becoming so complex that a single architect is not capable of covering the entire breadth of the required detailed knowledge areas. In those cases a team of architects is required, that is complementing each other in knowledge and skills. It is recommended that those architects have complementary roots as well; as this will improve the credibility of the team of architects.

Figure 3.4 shows that the scope of architects widely varies. The common denominator for all these architects is the bridge function between context and technology (or problem and solution). An architect needs sufficient know-how to understand the context as well as the technology, in order to design a solution, which fits in the context and is technical sound at the same time.

In general increasing the product scope of an architect coincides with an increase in people scope at the same time.

### 3.3 Education

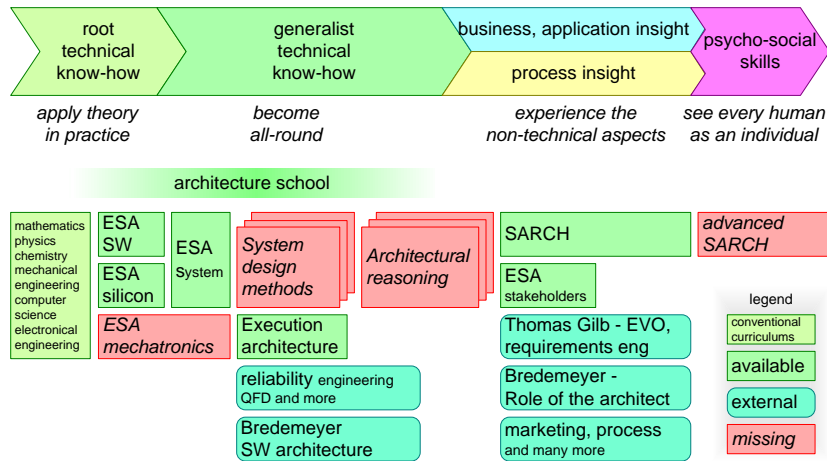


Figure 3.5: Proposed Curriculum for System Architects

A curriculum proposal for architects is shown in Figure 3.5. At the top of the figure the growth path of a system architect is shown. Below the courses or course subjects are shown which fit in the architect career path. Note that this is not a unified list for all architects. Instead it is a palette of courses, where the architect must select the courses which best fit his current needs. In color coding is indicated if courses are available internal or external.

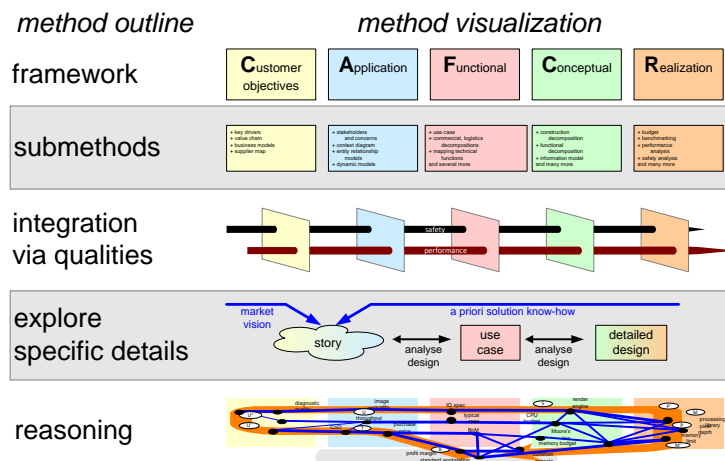


Figure 3.6: The outline of a CAFCR based architecting method

Figure 3.6 shows the overall outline of an architecting method, as it is being

used in the MOSAD or *Architectural Reasoning* course. The right hand side shows the visualization of the steps of the method. The *framework* is a decomposition into five views, the “CAFCR” model, *Customer Objectives*, *Application*, *Functional*, *Conceptual*, and *Realization* views.

Per view in the decomposition a collection of *submethods* is given. The collections of submethods are open-ended. The collection is filled by borrowing relevant methods from many disciplines.

A decomposition in itself is not useful without the complementing integration. *Qualities* are used as *integrating* elements. The decomposition into qualities is orthogonal to the “CAFCR” model.

The decomposition into CAFCR views and into qualities both tend to be rather *abstract*, *high level* or *generic*. Therefore, a complementary approach is added to *explore specific details*: story telling. Story telling is the starting point for specific case analysis and design studies.

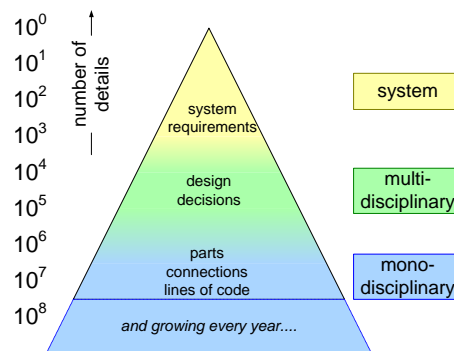


Figure 3.7: Connecting System Design to Detailed Design

These approaches are combined into a thread of *reasoning*: valuable insights in the different views in relation to each other. The basic working methods of the architect and the decompositions should help the architect to maintain the overview and to prevent drowning in the tremendous amount of data and relationships. The stories and detailed case and design studies should help to keep the insights factual.

The translation of system requirements into detailed mono-disciplinary design decisions spans many orders of magnitude. The few statements of performance, cost and size in the system requirements specification ultimately result in millions of details in the technical product description: million(s) of lines of code, connections, and parts. The technical product description is the accumulation of *mono-disciplinary* formalizations. Figure 3.7 shows this dynamic range as a pyramid with the system at the top and the millions of technical details at the bottom.

The combination of Figures 3.6 and 3.7 brings us to a very common organizational problem: the disconnect between customer oriented reasoning (breadth,

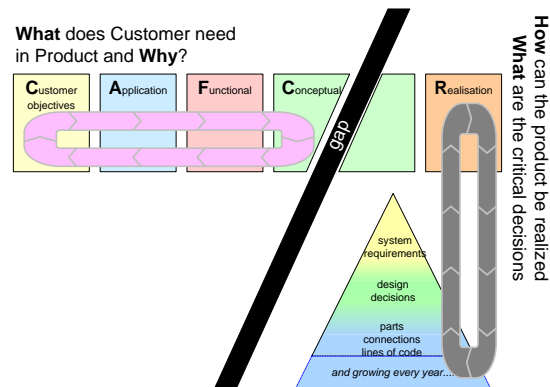


Figure 3.8: Organizational Problem: Disconnect

CAFCR) and technical expertise (depth, the mono-disciplinary area in the pyramid). Figure 3.8 shows this disconnect.

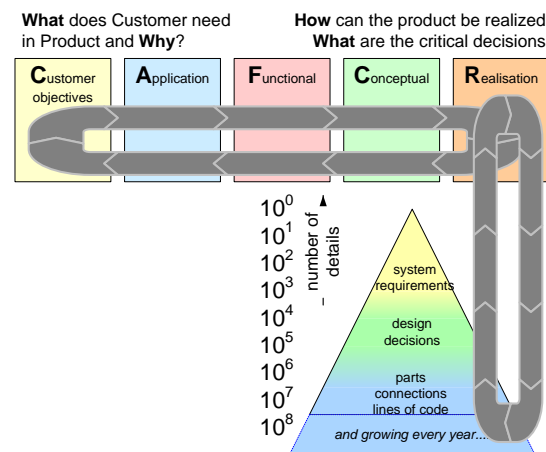


Figure 3.9: Architect: Connecting Problem and Technical Solution

Our definition of the work of an architect places this role as a bridge between these two worlds, as shown in Figure 3.9. In essence the architect must combine and balance breadth and depth iterations.

We should realize that this architect role is quite a stretching proposition. The architect is stretched in customer, application and business direction and at the same time the same architect is expected to be able to discuss technological details at nuts and bolts level. By necessity the architect will function most of the time at higher abstraction levels, time does and brain capacity don't allow the architect to spend all time at detailed design level. Figure 3.10 shows that different people fill

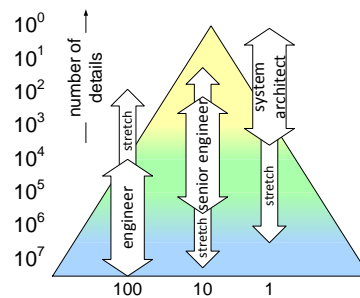


Figure 3.10: Major Bottleneck: Mental Dynamic Range

different spots in the abstraction hierarchies. For communication purposes and to get a healthy system design the roles must have sufficient overlap. This means that all players need to be stretched regularly beyond their natural realm of comfort.

The MOSAD course provides means to address:

- the breadth of systems architecting
- the depth of technological design
- the connection of breadth and depth

If we look back at the first editions of the MOSAD course, then we see that participants have the tendency to either go for breadth or for depth. But exploring both breadth and depth, and even more challenging connecting breadth and depth appears to be very difficult.

### 3.4 Nature

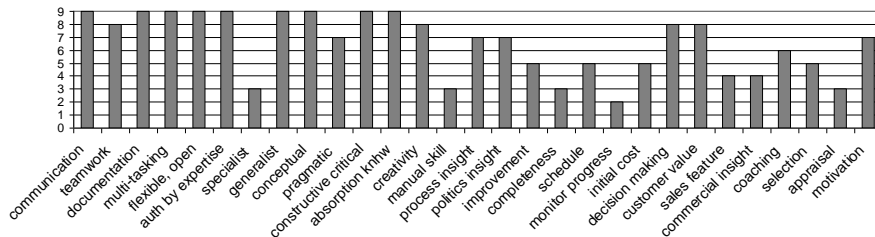


Figure 3.11: Profile of an "Ideal" System Architect

The profile of the "ideal" system architect shows a broad spectrum of required skills, as shown in Figure 3.11. A more complete description of this profile and the skills in this profile can be found at [11]. Quite some emphasis in the skill set is on *interpersonal skills*, *know-how*, and *reasoning power*.

This profile is strongly based upon an architecting style, which is based on technical leadership, where the architect provides direction (*know-how* and *reasoning power*) as well as moderates the integration (*interpersonal skills*).

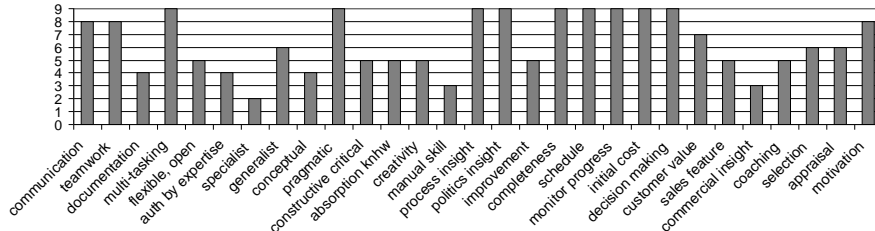


Figure 3.12: For Comparison: Profile of a Project Leader

The required profile is so requiring that not many people fit into it, it is a so-called **sheep with seven legs**. In real life we are quite happy if we have people available with a reasonable approximation of this profile. The combination of complementary approximations allows for the formation of architecture teams, which as a team are close to this profile.

For comparison the profile of a project leader is shown in Figure 3.12. A project leader is totally focused on the result. This requires project management skills, which is the core discipline for project leaders. The multi-tasking ability is an important prerequisite for the project leader. If this ability is missing the person runs a severe risk on a burn out.

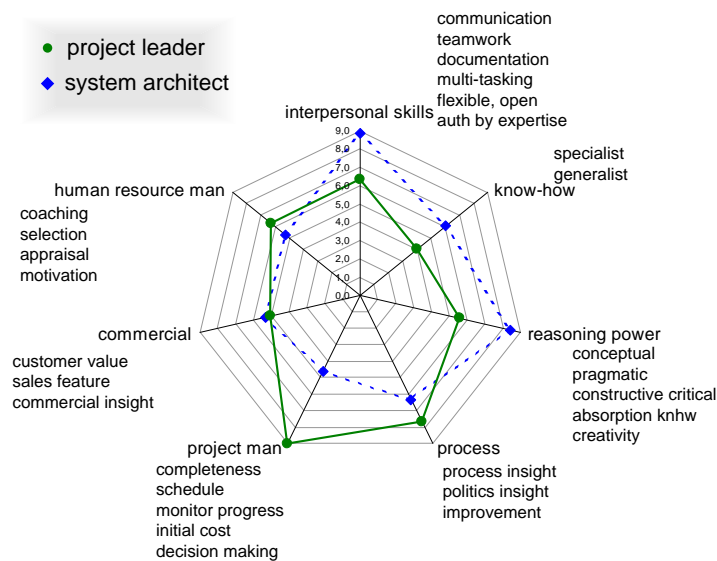


Figure 3.13: Project Leader versus System Architect

The comparison is further visualized in Figure 3.13, where the more detailed skills from Figures 3.11 and 3.12 are grouped together.

- Generalist
- Multi-tasking
- Authority by expertise
- Constructive critical
- Balance between conceptual and pragmatic

Figure 3.14: Most Discriminating Characteristics

In practice the characteristics shown in Figure 3.14 are quite discriminating when selecting (potential) system architects: The first reduction step, when searching for architects, is to select the generalists only. This step reduces the input stream with one order of magnitude. The next step is to detect those people which need time and concentration to make progress. These people become unnerved in the job of the system architect, where frequent interrupts (meetings, telephone calls, people walking in) occur all the time. Ignoring these interrupts is not recommendable, this would block the progress of many other people. Whenever these

people become system architect nevertheless they are in sever danger of stress and burn out, hence it is also the benefit of the person itself to fairly asses the multi-tasking characteristic.

The attitude of the (potential) architect is important for the long term effectiveness. Roughly two attitudes can be distinguished: architects that ask for formal power and architects that operate on the basis of build-up authority. Building up authority requires know-how and visible contribution to projects. We have observed that architects asking for formal power are often successful on the short term, creating a single focus in the beginning. However in the long run the inbreeding of ideas takes its toll. Architecting based on know-how and contribution costs a lot of energy, but it pays back in the long term.

The balance between conceptual thinking and being pragmatic is also rather discriminating. Conceptual thinking is a must for an architect. However the capability to translate these concepts in real world activities or implementations is crucial. This requires a pragmatic approach. Conceptual-only people dream up academic solutions.

### 3.5 Experience

The effectiveness of an architect depends on experience. In all years of being an engineer, designer and architect, a lot of different needs in different contexts with different solutions with different complicating challenges pass by. If all these events are processed by the (potential) architect, then a frame of reference is created that is very valuable for future architecting work.

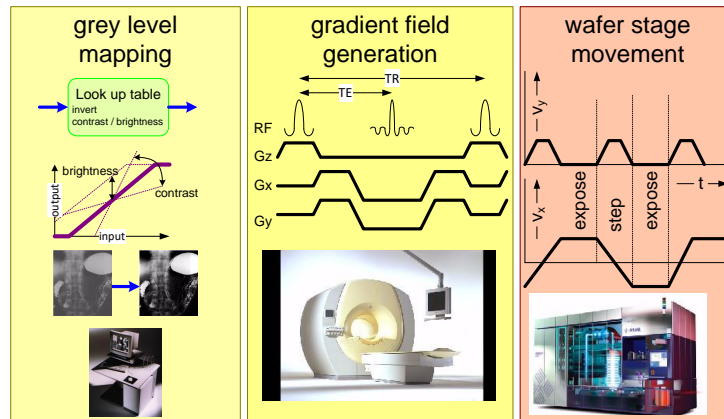


Figure 3.15: Example: Trapezoid Pattern

In this section we will illustrate the experience factor by means of a few architecture patterns that repeatedly popped up in completely different domains. For this purpose we look at the *Trapezoid Pattern*, as shown in Figure 3.15. One of the very common technical problems is the actuation by software of some physical entity, for instance for positioning, moving or displaying. In these cases the software often has to create set-points for one parameter, where this parameter is constant at different levels for some time and switches linearly from one level to another level. For instance, a sample table is at a given position (constant), moves with a constant velocity to the next position, and then stays at the next position for some time (constant). This same behavior is also present in the actuation of gradient fields in MRI scanners, and in the grey level mapping in imaging displays (although the last example uses grey levels as running parameters instead of time).

In the system a chain of transformations takes place to get from a high level software representation towards an actual physical behavior by physical objects. Figure 3.16 shows such a chain of three steps: *computation*, *conversion*, and *actuation*. Note that this chain is often more complex in real systems with more software steps (controller algorithms, corrections), more electronic steps (controllers, amplifiers), and more mechanical steps (motors, transmission). The high level software representation that is the starting point is unconstrained (high precision in time as well as in value). The most common representation is break-point based: the coordinates,

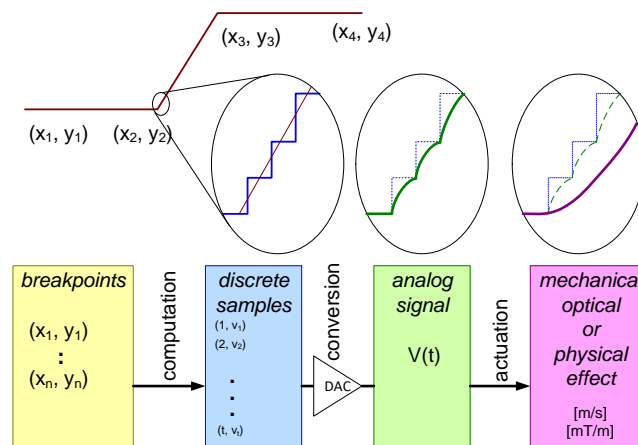


Figure 3.16: From SW input to physical Effect

where the running parameter changes the linear behavior, are specified.

The conversion and actuation steps have their own particular transfer functions. These steps may introduce additional delays, noise, variations et cetera. The virtual model in the high level software does not take this into account or makes (calibrated) assumptions.

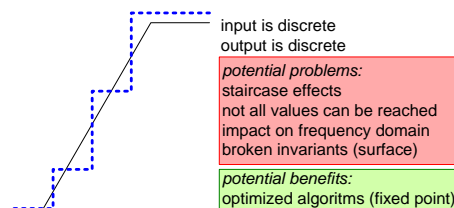


Figure 3.17: Discretization effects

The *computation* step transforms the unconstrained representation into a constrained sampled list of values. This transformation is a discretization in two directions: time and value, see Figure 3.17. This discretization may introduce system level problems:

**Staircase effects** the linear shape is approximated by many staircase-like steps.

The question is how this software output is transformed into the actual physical actuation and if artifacts will be observable in the physical performance.

**Not all values can be reached** . Normally the digital to analog conversion is a bottleneck in the values that can be reached. This conversion can be very much limited in low cost solutions (8-bits, 256 values) to limited (16-bit, 65536 values). The time-values are also limited, varying from sub-microsecond

for more expensive solutions to milliseconds for simple low-cost controls. The consequence of this limitation is that the physical reality may differ in a systematic way from the virtual model in the high level software. For example the high level software may have determined that at moment  $t = 3.14159$  the system should be at position  $x = 2.718281$ , while actually the system is controlled to stop at  $t = 3.1$ ,  $x = 2.7$ .

**Impact on frequency domain** The staircase approximation of linear behavior introduces many higher frequencies in the frequency domain. Many of the higher frequency artifacts are filtered out in the analog and physical part of the chain. However, due to aliasing-like problems the system performance might degrade in unexpected ways.

**Broken invariants (surface)** The high level software model in many systems is based on invariants. For instance, if we control velocity linear, then we expect that we know the position as the integral of velocity. Discretization, at lower software level, will violate the higher level assumption. If the model assumes we move with  $v = 3.14159m/s$ , while we actually move with  $v = 3.1m/s$ , then the position will deviate significant. Interestingly, the low level software can compensate for this error by modulating the value: 58% of the time  $v = 3.1m/s$  and 42% of the time  $v = 3.2m/s$ . These solutions work, but introduce again their own next level of problems and artifacts. In this example the frequency of the modulation may introduce unexpected physical behavior, such as vibrations.

A priori use of the need for discretization can also turn into a benefit. Especially the consequent use of integer representations (with some pragmatic normalization, such as  $255 = 5V_{olt}$ ) reduces processor load, memory use and may increase system performance.

Discretization problems, the artifacts introduced by discretization, the measures against artifacts are also universally applicable. However, the exact consequence and the right countermeasure are domain dependent.

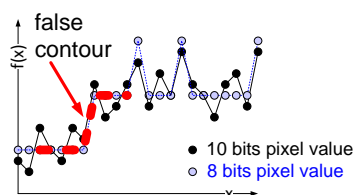


Figure 3.18: Example of Discretization Problem

As example of discretization problems Figure 3.18 shows a typical image quality problem that popped up during the integration phase of a Medical Imaging Workstation.

The pixel value  $x$ , corresponding to the amount of X-ray dose received in the detector, has to be transformed into a grey value  $f(x)$  that is used to display the image on the screen. Due to discretization of the pixel values to 8 bits *false contours* become visible. For the human eye an artefact is visible between pixels that are mapped on a single grey value and neighboring pixels that are mapped on the next higher grey value. It is the levelling effect caused by the discretization that becomes visible as false contour. This artefact is invisible if the natural noise is still present. Concatenation of multiple processing steps can strongly increase this type of artifacts.

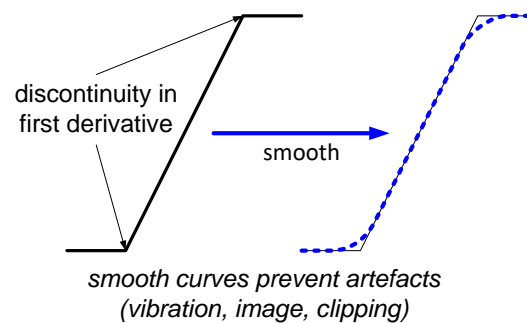


Figure 3.19: Example of Generic Smoothing Consideration

An example of a pattern that builds further on this transformation chain is shown in Figure 3.19. Physical systems in general start to show artifacts with discontinuous inputs. The linear approximation used in the trapezoid pattern has a discontinuity in the derivative. For example, if we control velocity, then the acceleration jumps at the break-point. A solution for this discontinuity is to *smooth* the input function, for instance by a low-pass filter. Note that most analog and mechanical systems are already natural low-pass filters. Despite the low-pass characteristic of the later part of the chain artifacts might still be induced by the discontinuity. These remaining artifacts can be further removed by using an explicit low-pass filter in the high level software model. Again this is an example of a pattern that is universally applied in multiple domains.

The example showed a small subset of patterns that an architect experiences. This subset as it has been discussed here is highly technical. However, in real life technical patterns and organizational patterns are experienced concurrently. For example in the trapezoid example also a number of organizational patterns pop up, related to mono-disciplinary experts and multi-disciplinary design, and system integration.

In Figure 3.20 the career of an architect is shown with the repeated encounters of patterns in different products and in different environments. We estimate that an experiences architect encounters (and files and uses) thousands of patterns. All

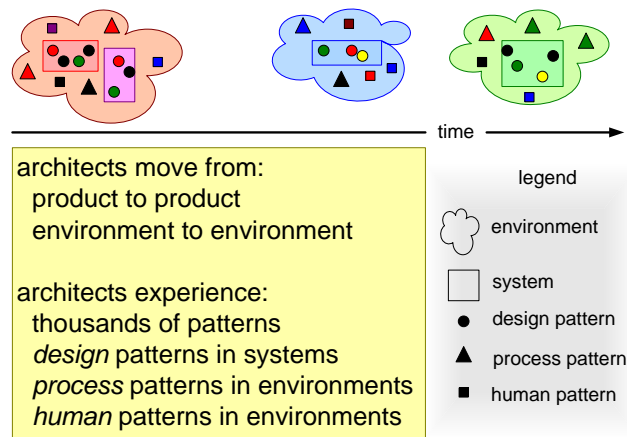


Figure 3.20: Architects Collect a Rich Set of Patterns

these patterns form a frame of reference for the architect as an individual. This frame of reference helps the architect to assess new architectures very quickly. Potential problem areas are identified and design issues are weighted very fast, thanks to this frame of reference.

### 3.6 Environment

The business process for an organization which creates and builds systems consisting of hardware and software can be decomposed in 4 main processes as shown in figure 3.21. This process decomposition model is more extensively discussed in[9].

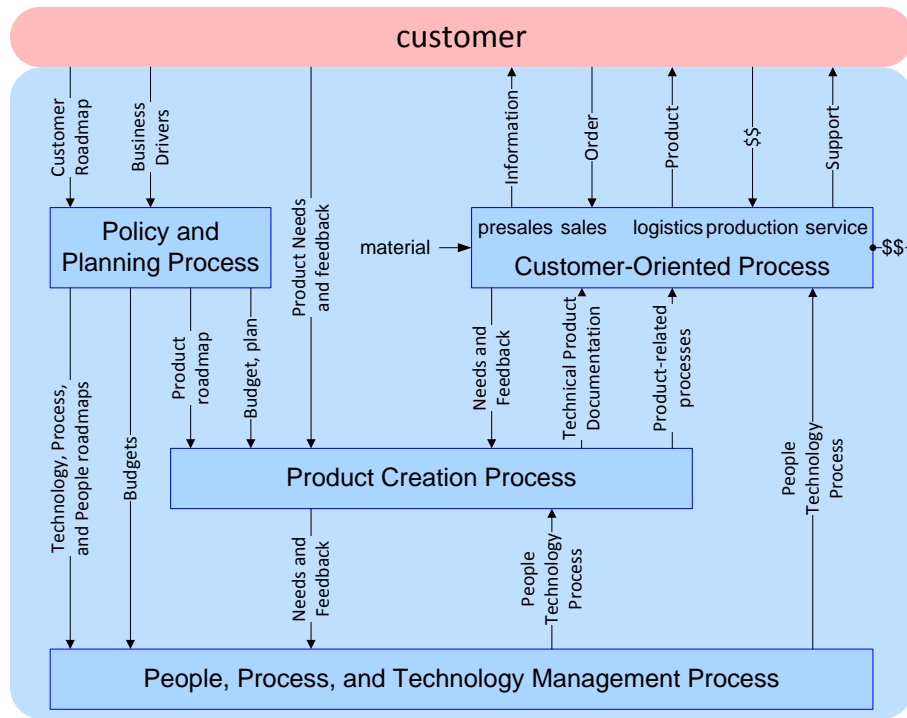


Figure 3.21: Simplified decomposition of the Business

The decomposition in 4 main processes leaves out all connecting supporting and other processes. The function of the 4 main processes is:

**Customer Oriented Process** This process performs in repetitive mode all direct interaction with the customer. This primary process is the cash flow generating part of the enterprise. All other processes only spend money.

**Product Creation Process** This Process feeds the Customer Oriented Process with new products. This process ensures the continuity of the enterprise by creating products which enables the primary process to generate cash flow tomorrow as well.

**People and Technology Management Process** Here the main assets of the company are managed: the know how and skills residing in people.

**Policy and Planning Process** This process is future oriented, not constrained by short term goals, it is defining the future direction of the company by means of roadmaps. These roadmaps give direction to the Product Creation Process and the People and Technology Management Process. For the medium term these roadmaps are transformed in budgets and plans, which are committal for all stakeholders.

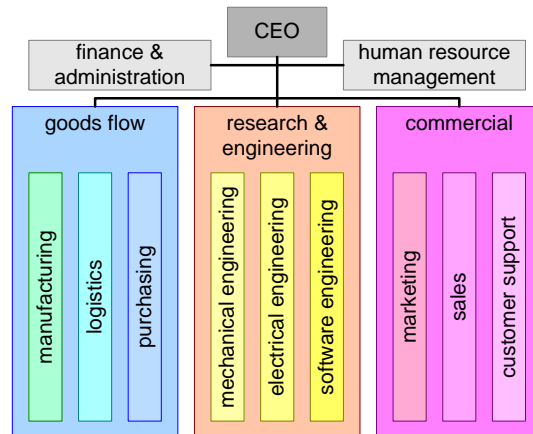


Figure 3.22: Line Organization Stovepipe

The challenge for companies is to organize themselves in a way that support these 4 different types of processes. Rather common is that the People and Technology Management Process is mapped on the line organization, see Figure 3.22. This figure also shows a common problem of hierarchical organization structures: the organizational entities become (over)specialized stovepipes.

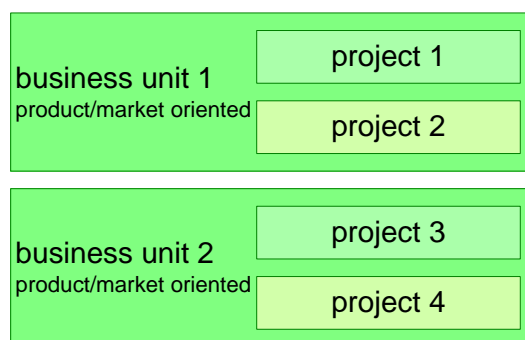


Figure 3.23: Business Organization Stovepipe

The *Product Creation Process* maps often on a business oriented project organi-

zation, as shown in Figure 3.23. The stovepipe problem is here also present, although the stovepipes are now in the product/market direction.

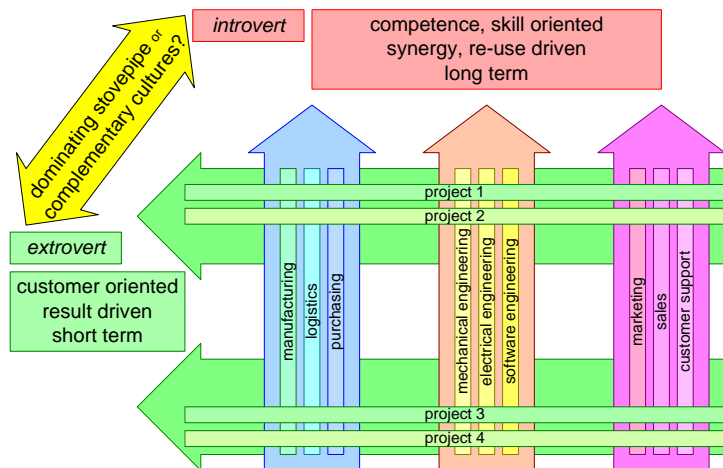


Figure 3.24: Different Concerns

The combination of both organization models results in a matrix organization, where the two types of organizations have different concerns. The line organization is competence and skill oriented, looking for synergy and re-use opportunities. The line organization typically has a long term focus, but an introvert perspective. The business organization is customer oriented and result driven. The business organization typically has a short term focus, but an extrovert perspective.

Figure 3.25 positions the *System Architecture Process* in the simplified process decomposition. The System Architecture Process bridges the Policy and Planning Process and the Product Creation Process. The roadmaps made in the policy and planning process are the shared understanding of direction of the company:

- It positions the products in the market and within the product portfolio.
- It shows the relations between products, such as re-use of technology.
- It positions the product in the technology life-cycle.
- It relates products and technology to the (long lead) development of people and process

The System Architecture Process is the process that:

- Gathers input for the Policy and Planning Process
- Brings in technical overview and common sense in the Policy and Planning Process and the Product Creation Process
- Transfers the intention of the Policy and Planning Process into the Product Creation Process

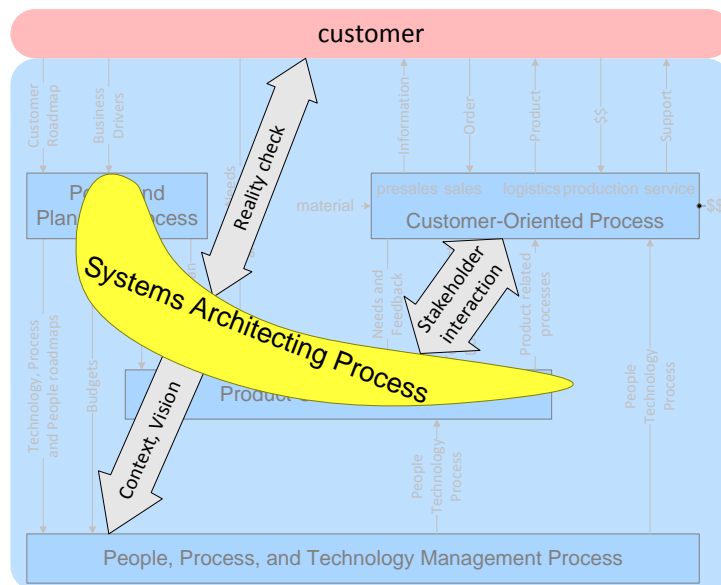


Figure 3.25: Positioning System Architecting

- Performs the system level Requirement analysis, Specification, Design and Verification
- Maintains the consistency, integrity and balance.

systems engineering as discipline  
 job rotation  
 stimulate architect exposure  
 stretch all engineers  
 cultivate customer & market oriented culture  
 share and invest in future exploration and vision

Figure 3.26: What Can We Do to Improve the Environment?

Until now we have sketched the organizational and process environment in which the system architect operates. A complex environment that is full of human factors, such as conflicting interests and complementing (or opposing?) characters. The natural growth direction in this environment is specialization. In some organizations the security or standardization efforts hurt the architecting effectiveness. For example, we have seen organizations where customer key drivers, cost of ownership models, and market roadmaps are marketing confidential. The gap as

described in Figure 3.8 is here imposed by the organization.

Figure 3.26 shows what we can do to improve the environment from system architecting perspective.

**Systems engineering as discipline** Conventional disciplines are technology oriented, for instance: mechanical, electrical, and software engineering. However, systems engineering has grown into a discipline itself. Most organizations have a severe lack of systems engineers and systems architects. Organizational ownership for systems engineering as a discipline counter-balances the natural tendency towards specialization.

**Job rotation** is one of the means to broaden employees. The cultivation of a systems attitude requires such a broadening, it is a prerequisite to become systems engineer

**Stimulate architect exposure** to help them overcome their introvert nature and to help them bridge the gap between managers and architects.

**stretch all engineers** The broadening mentioned before should not be limited to (potential) system architects. The extremely challenging job of a system architect becomes somewhat more feasible if the engineers are at least system-aware.

**cultivate customer and market oriented culture** Especially in large organizations the distance from local organizational concerns to customers and market can become large. System architects suffer tremendously from introvert organizations, because the architect has to connect the customer and market needs to technological decisions.

**share and invest in future exploration and vision** Good architects base their work on a vision. Some investment is needed to create a vision and to keep the vision up-to-date. A vision becomes much more powerful if it is shared throughout the organization.

### 3.7 Discussion and Conclusions

This paper was triggered by the not yet satisfactory results of our newly developed MOSAD course. Analysis of the critical success factors for system architects provides us with the following insights:

- Only a limited set of technical educated people have a personality profile (the *nature* component) that fits with the architecting role.
- System architecting education for people that do **not** fit in this architect profile is, nevertheless, a good investment. System aware designers ease the job of the system architect.
- Environmental issues, such as organization and processes, have a big impact on the effectiveness of architects.
- Architects need to be stimulated and supported to break through roadblocks imposed by the environment.
- To integrate and use multi-disciplinary design techniques a broad frame of reference is needed. Such a frame of reference helps to position, relate and weight issues, and to identify risks. Without the ability to quickly determine value, relevance and criticality, designers drown in the practical infinite space of problems and solutions.
- A frame of reference grows over time and is the result of experience. This process can be supported by explicit reflection, for instance triggered by a mentor or intervention by peers.

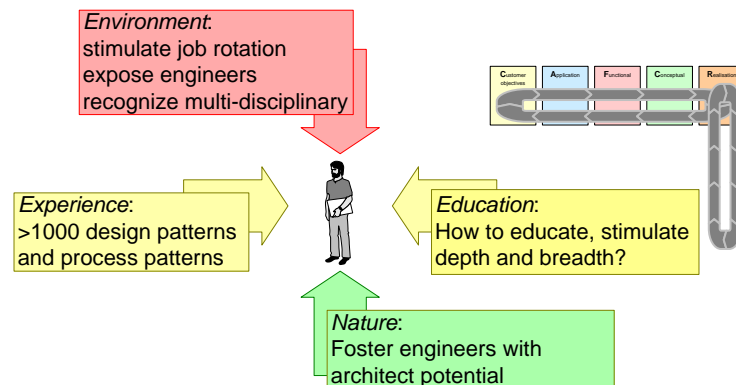


Figure 3.27: Conclusion

Figure 3.27 summarizes the conclusions:

**Education** How do we stimulate and educate breadth and depth synthesis?

**Nature** People with architecting genes are scarce; We have to foster and stimulate those people that fit in the architecting profile.

**Experience** plays a very critical role in cultivating architects. Good architects have a very rich frame of reference with thousands of patterns.

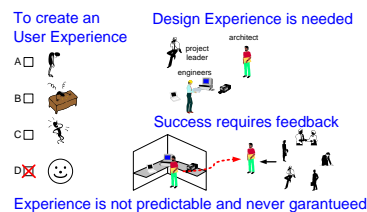
**Environment** has a big impact on architect effectiveness. Stimulation of job rotation helps to enrich the frame of reference. By exposing engineers to multi-disciplinary aspects the awareness for system issues increases. The environment (management, rewarding system) must recognize the value of multi-disciplinary design.

### **3.8 Acknowledgements**

Louis Stroucken detected a painful copy-paste error and provided other useful feedback.

## Chapter 4

# Architecting for Humans; How to Transfer Experience?



### 4.1 Introduction

Many modern appliances cause an alienated feeling for (less-technical) consumers. Figure 7.1 shows a multiple choice set of feelings for programming the well known Video Cassette Recorder (VCR) or its later successor the Personal Video Recorder (PVR). This task of programming the VCR is often delegated to the family member with sufficient technical feeling.

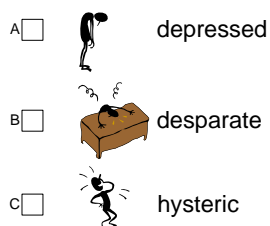


Figure 4.1: Did you ever program a Video Recorder (VCR or PVR)?

A long lasting process is performed to come from some consumer need to a manufacturable, salable product. This product creation cycle is shown in figure 7.2.

It starts with a product manager, who perceives a product opportunity as a need from the user. The product manager formulates the product requirements, which are used by a development team, consisting of engineers, architect and project leader, to design a product. The final result of the engineering effort is a "product documentation", which is used by manufacturing to produce the product and by sales to sell the product. Via the retail channels the product finally arrives at the consumer.

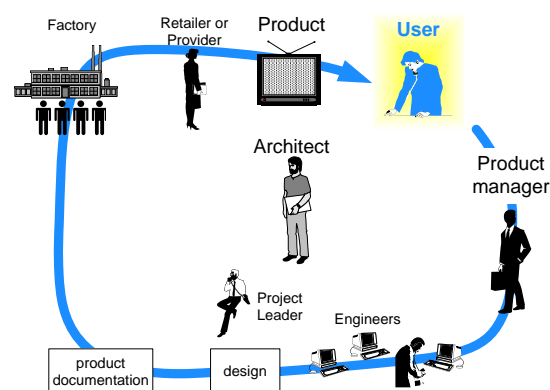


Figure 4.2: Product Creation Cycle

The word "experience" in the title of this article is used in a double meaning:

- the feelings and emotions an user experiences when using the system
- the accumulated skills and know-how of working many years in the domain

Both concepts of experience share the difficulty or in fact impossibility of transferring experience from one human to the next. Figure 4.3 visualizes these 2 forms of experience.

The experience of the human using a new product, resulting from an engineering activity, is determined by emotions, feelings, opinions, et cetera. At the other hand engineering a product from available technologies is in many aspects a very SMART activity. See also [10] for a further discussion on the relation between "fuzzy" user needs and SMART engineering. Figure 7.3 visualizes the gap between the user experience and the engineering world, which is to bridged by an architecting effort.

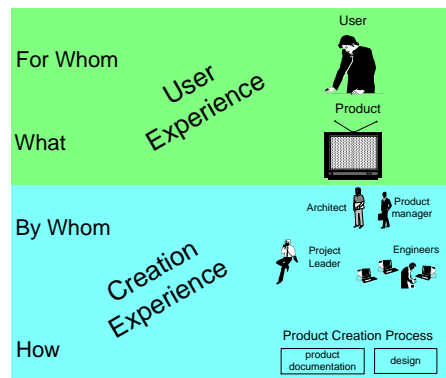


Figure 4.3: 2 Levels of Experience

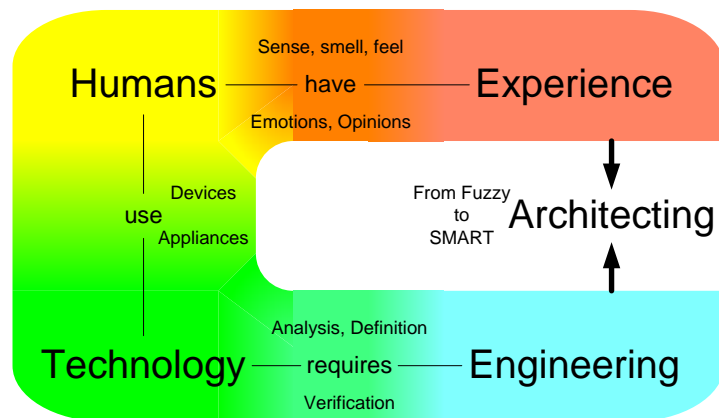


Figure 4.4: Bridging the gap between Experience and Engineering

## 4.2 User Experience

As an example of user experience Time shift recording is used. Figure 8.9 shows the concurrent activities that occur when straightforward time shifting is used. In this example the user is watching a movie, which is broadcasted via conventional means. After some time he is interrupted by the telephone. In order to be able to resume the viewing of the movie he pauses the viewing, which starts invisible the recording of the remainder of the movie. Sometime later he resumes viewing where he left off, while in the background the recording of the not yet finished movie continues.

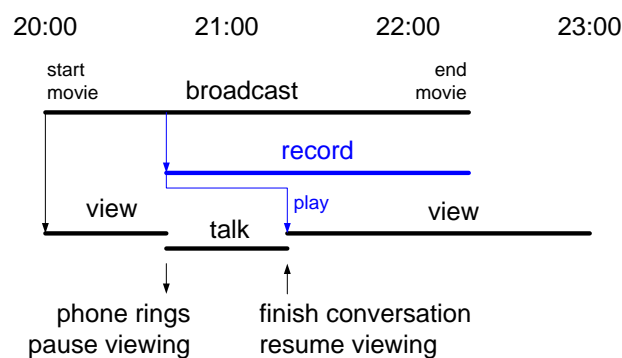


Figure 4.5: Example Time Shift recording

In this simple form (pause/resume) this function provides freedom of time to the user. This appears to be very attractive in this interaction modus. However when such an appliance is designed limits out of the construction world pop up, which intrude in the user experience. Table 4.1 shows a number of construction limits, which are relevant for the external behavior of the appliance.

- number of tuners
- number of simultaneous streams (recording and playing)
- amount of available storage
- management strategy of storage space

Table 4.1: *Construction limits intrude in Experience*

Construction limits, but also more extensive user stories, see figure 4.6, show how the intrinsic simple model can deteriorate into a more complex interaction model. Interference of different user inputs and interference of appliance limitations compromise the simplicity of the interaction model.

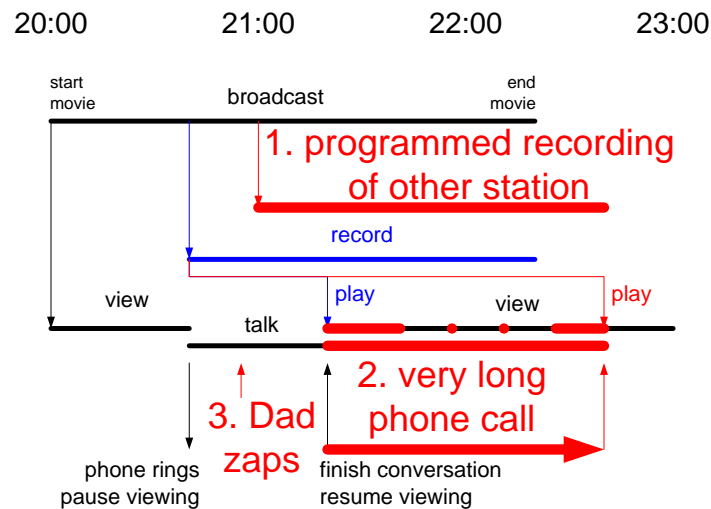


Figure 4.6: What if?

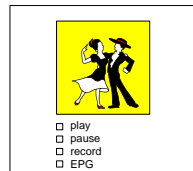
The story behind figure 4.6 is that Sharon, mother of 15-year old Brigit, is watching the latest Meryl Streep movie on television. This entertainment is interrupted by a phone call from her sister. Sharon pauses the movie and talks extensively with her sister. Five minutes later dad, Bob, walks in the room and zaps to CNN, to watch the latest developments. At 9 o'clock a recording should start of a soap series, programmed by daughter Brigit. 9:15 Sharon says her sister good bye and presses resume to continue with her Meryl Streep movie.

The big questions in this story is: *What is recorded when?* and *Who is able to watch the desired content later this evening?*. Most Personal Video Recorders have only one tuner and will therefore not support the three persons satisfactory.

In the Post doctoral education for computer science designers at the technical university Eindhoven, the students have to design such a time shift appliance. In the function of "requirement expert" I was involved in this design workshop. The initial effort of the students was heavily focused on creating a requirement specification, full with tables defining requirements. This thick stack of paper did not really help the students to understand the essence of the appliance, nor did it help to identify the critical or difficult issues. After challenging them the students build a functioning prototype on a PC, which immediately surfaced a number of critical issues and enabled discussion and feedback on the user interaction model, see figure 4.7.

The user experience is influenced by many factors, ranging from environmental factors, such as social status, location and time to personal factors, such as education, preferences and physical status. This wide variation of influencing

Visual Basic Prototype:  
enables "experiencing"



Requirements specification  
Many tables, mostly addressing details

- 2.1.1 Real-time data requirements
- 2.1.2 Implementation detail
- 2.1.3 Non-real time data requirements

2.1 Software Requirements	
2.1.1 Real-time data requirements	<p>2.1.1.1 Access to the non-real-time data must be done in such a way that it does not interfere with the real-time data</p> <p>2.1.1.2 There must be no disruptions in output of video signal during the operation of VCR</p> <p>2.1.1.3 Requirements for non-real-time data is less than 100ms (the time for writing a block on HDD) or 2000 of non-video data</p>
2.1.2 Implementation detail	<p>2.1.2.1 Management of VCR content must only be possible through the VCR in order to prevent unauthorized access to content of VCR</p> <p>2.1.2.2 Visual feedback is provided to the user via On-Screen Display</p>
2.1.3 Non-real time data requirements	<p>2.1.3.1 User input is processed via the PC</p> <p>2.1.3.2 User must be able to pause and resume a file played from HDD, while (s/he is watching it</p> <p>2.1.3.3 User can jump forward and backward to a file from HDD, during watching of this file</p> <p>2.1.3.4 Names of files should be derived from the information from the VCR (name of the program to be recorded, time and date of recording)</p>

Figure 4.7: OOTI workshop 2001

factors is shown in figure 4.8.

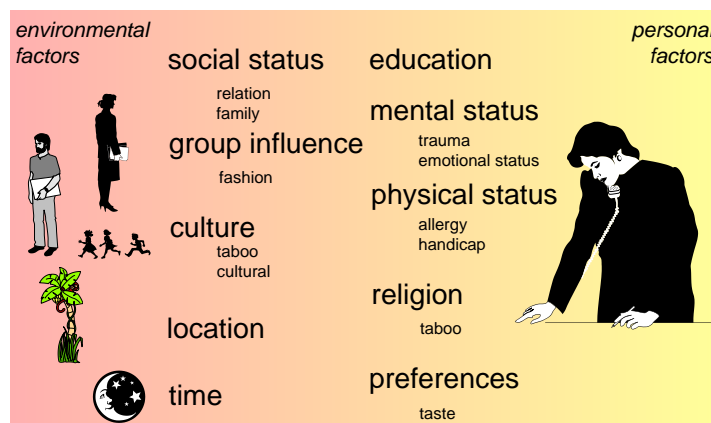


Figure 4.8: Factors influencing the User Experience

The challenge is to make the user experience more tangible, for instance by "SMART"ening the experience. Table 4.2 indicates what we would like to do with a "SMART"ened experience.

A consequence of all factors which determine the user experience is that the experience space is in practice infinitely large. The size of this experience space is the product of all users and all values that every influencing factor can have. Figure 4.9 shows for only a few influence factor the size explosion of this experience space.

Although the infinite size of the experience space might suggest the impossibility to design good products, it is not that bad:

- define
- measure
- predict
- verify

Table 4.2: *How to "SMART"en Experience?*

People	Number of People on earth	$O(10^9)$ *
Time	Human lifespan in seconds	$O(10^9)$ *
Location	Square meters of planet earth	$O(10^{14})$ *
...	...	...
Size of experience space		$\infty$

Figure 4.9: Infinite Experience Space

*It is not that bad :-)*

*Many nice and successful products exist!*

One of the important means to achieve successfully products is the abundant use of feedback. Figure 4.10 shows some important aspects of obtaining feedback; get architects and designers out of the development laboratory; use short development cycles and observe and listen to users.

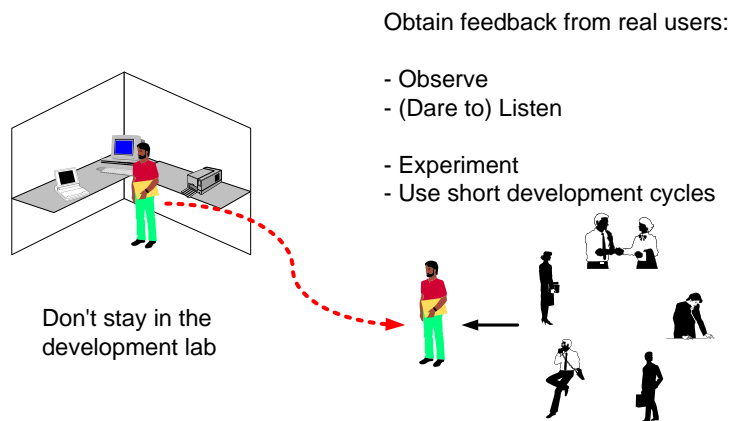


Figure 4.10: Key Success Factor: Feedback

## 4.3 Engineering

The world of engineering and construction is full of technologies and tools. Figure 4.11 shows some commonly used elements of this world.

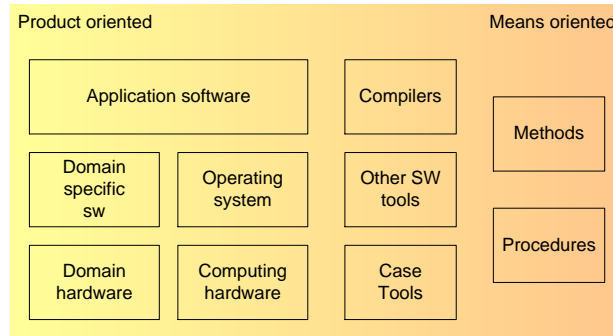


Figure 4.11: The world of the construction

The engineers are educated in construction disciplines: how to apply technologies to realize a solution which fits the specified requirements. Table 4.3 shows some of the disciplines in the education of an engineer.

- Programming languages
- Operating systems
- Algorithms
- Data structures
- Formal specification and verification techniques
- Analysis, simulation techniques

Table 4.3: *Engineers are educated in construction disciplines*

Product Creation is much more than engineering only. Engineering is an important part of product creation, which enables the engineers to re-use methods, tools et cetera (see figure 4.12 for more detail). However on top of engineering also creativity is needed, where creativity is based on diverse sources as intuition, lateral thinking, trial and error et cetera. The engineering know-how can be taught in courses, while the creativity is developed (and should be latent available) mostly by experience.

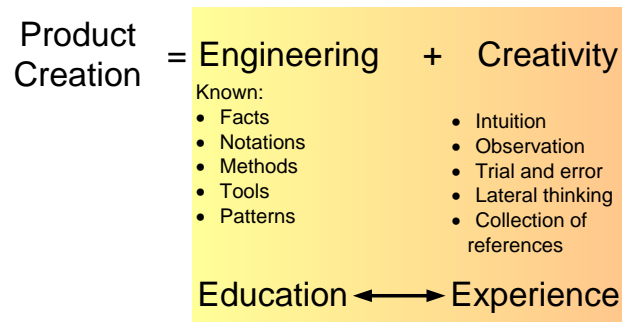


Figure 4.12: Product Creation is much more than Engineering

## 4.4 Education

The understanding that product creation is a combination of engineering and creativity helps to formulate an educational curriculum for architects and designers.

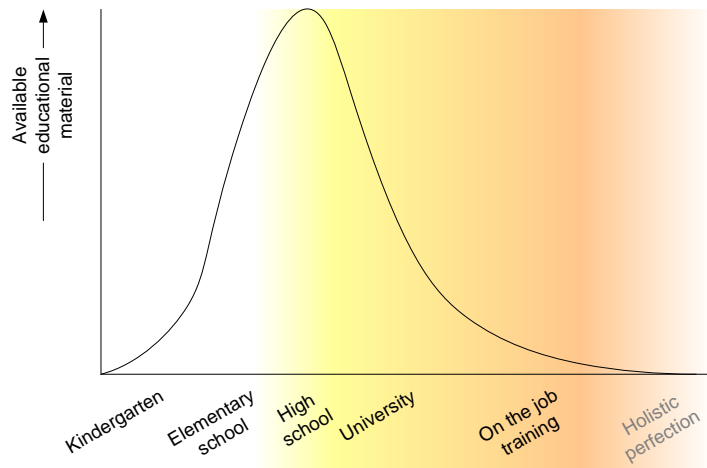


Figure 4.13: Educational Material per education stage

Figure 4.13 shows that for education at schools and universities quite a lot of educational material is available. However the more advanced education becomes the less material is available.

Do	Exercise	Practical training	apprentice-ship	Peer coaching
Interact and Listen	Lectures: Explain Show examples		Seminars Workshops Conferences	
Read	Handbook Course material		Magazines Journals	
time →				

Figure 4.14: Changing Education model in time

The education can be decomposed in 3 types of learning: *doing*, *reading* and *interacting and listening*, see figure 4.14. This figure also shows that the contents for these categories changes over time.

The *doing* during school and university is mostly practising by making exercises, this evolves into practical training, then into apprenticeship and finally it becomes peer coaching.

The *interacting and listening* and *reading* evolve from a preprogrammed fashion at school and university into a selection of seminars, workshops, magazines et cetera.



Figure 4.15: Increasing Initiative required

Figure 4.15 annotates the education lifecycle with the characteristics. The early lifecycle is characterized by a limited scope, well organized, well specified subjects and involvement of a few (if any at all) stakeholders. At the later stages the characteristics are more or less the opposite: a large scope full of uncertainty and with many stakeholders. In the later stages the initiative for further education should come from the employee itself, no well organized curriculum exists anymore.

- Awareness of engineers of human aspects
- Active personal development drive of engineers
- Awareness of managers of education models
- Active motivation by managers

Table 4.4: *Prerequisites for continuous successful product creation*

Table 4.4 shows the prerequisites to be successful in product creation on a continuous basis. Both the manager as well as the employee should be aware of the need for further personal development, where the manager should stimulate and the engineer should have a personal drive.

## 4.5 Conclusion

Figures 4.16 and 4.17 summarize the article.

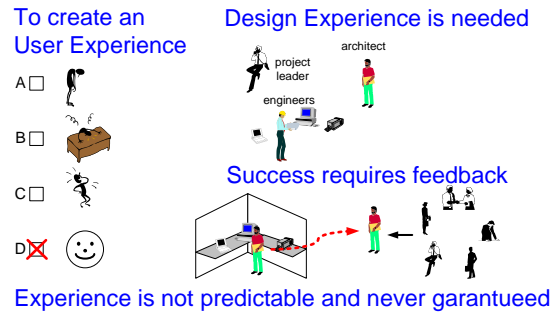


Figure 4.16: Architecting for Humans

User experience is never predictable, nor is it possible to guarantee an experience. Using methods derived from design experience and applying lots of feedback increases the chance on success.

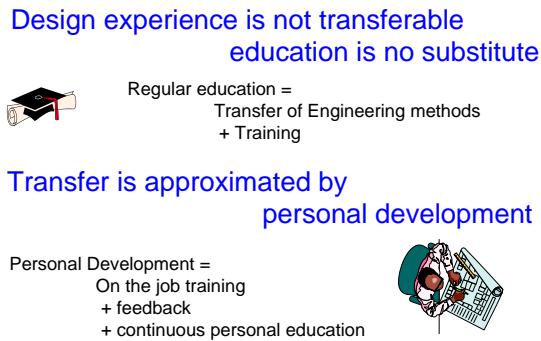


Figure 4.17: Experience Transfer

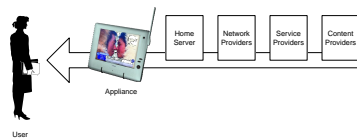
Design experience itself is not transferable, education is a means which can enable in potential good engineers to build up experience faster. This requires a lot of practical training. Later on this engineer must continue his personal development, by means of on the job training, feedback and peer coaching.

## **4.6 Acknowledgements**

Daniel Malacarne provided feedback on a number of figures.

## Chapter 5

# From the soft and fuzzy context to SMART engineering



### 5.1 Introduction

Engineering education and process improvement actions always stress the importance of "SMART"ness of requirements. Table 5.1 shows the meaning of SMART. The original use of this acronym was by George T. Doran, in an article about management goals and objectives [4]. Today the acronym is mostly used to stress the **specificity** and **measurability**, the "ART" part is used in many variations as shown in this list.

- Specific
- Measurable
- Assignable (Achievable, Attainable, Action oriented, Acceptable, Agreed-upon, Accountable)
- Realistic (Relevant, Result-Oriented)
- Time-related (Timely, Time-bound, Tangible, Traceable)

Table 5.1: *The meaning of SMART*

It is a sound advice to write product specifications with the SMART acronym in mind, every violation is a potential problem. However it is also important to try

to capture and communicate the understanding on which these smartened specifications are based. This understanding often deals with much more fuzzy issues.

By following one example it is shown which fuzzy inputs are provided in the beginning, what the different stakeholder needs are and how these inputs can be transformed into more SMART specifications.

## 5.2 Case description

Figure 5.1 shows the type of products used as an example: a *Mobile Display Appliance* and *Mediascreen*. The function of these products can vary from portable television to home control to web browser et cetera.

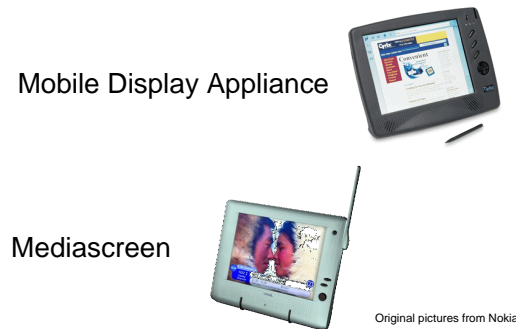


Figure 5.1: What are the requirements for these products?

Characteristic of these kind of products is that the functionality is typical a function of the appliance itself plus functions and content provided by the context. Figure 5.2 shows the total chain of systems which can be involved in the functionality.

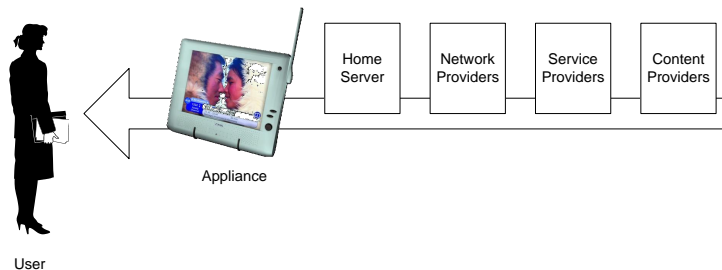


Figure 5.2: User access point to long food-chain

The architect of this appliance will receive many fuzzy expectations as inputs and for some parts he will get SMART descriptions. Figure 5.3 shows the fuzzy input as clouds and the SMART input as rectangles. Clearly a significant amount of fuzzy input is provided.

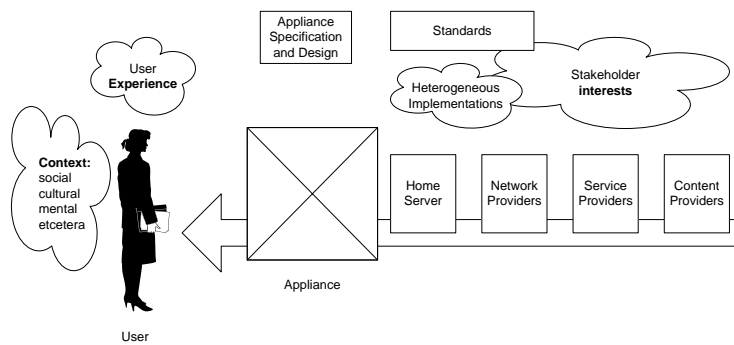


Figure 5.3: "Fuzzy expectations" and "SMART descriptions"

### 5.3 Why SMART?

The supply chain stakeholders of the design process are shown in figure 5.4. The product Creation Process takes care of the decomposition of the system in subsystems and components as well as the integration and test of the system. Later during manufacturing the components and subsystems are ordered from suppliers and assembled into a system. Finally the sales channels sell the systems and deliver the systems to the customers.

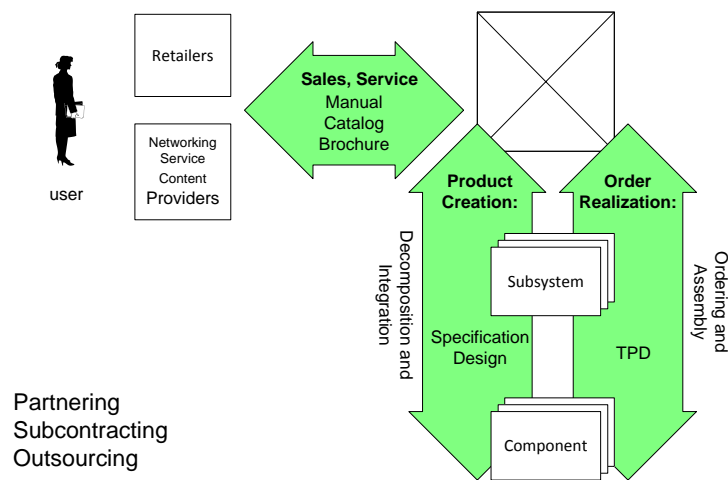


Figure 5.4: Supply Chain Stakeholders

All the stakeholders involved in this supply chain need specific and verifiable data to order, assemble, test and sell the product. (Did you ever try to tell a sales manager: "don't worry the product will be fast, will have a nice image quality and it will be very fashionable", without any further hard facts?)

Figure 5.5 shows the problem statement by visualizing all the fuzzy needs at the one hand and the SMART facts at the other hand.

Figure 5.6 sharpens the problem statement by showing the fuzzy elements playing mostly in a creative world (imagination) and the formalizations often used to make things work in a supply chain environment.

One very specific stakeholder is the supplier. Often outsourcing or purchasing processes are highly formalized, to prevent problems. Figure 5.7 shows this relationship, which often causes redundant specifications at the interface (one from the integrator point of view and one from the supplier point of view). The final formalization is laid down in a contract.

To make this relationship work it is important that the integrator has know-how and understanding of the supplier and vice versa, as shown in figure 5.8.

The decomposition and integration of the system requires SMART data at

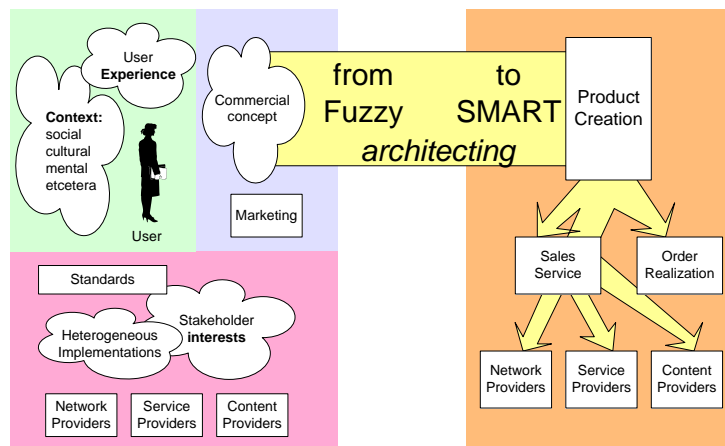


Figure 5.5: Problem Statement

all aggregation levels, both for product creation as well as for the supply chain. Figure 5.9 shows the functions in both processes, where SMART data are important.

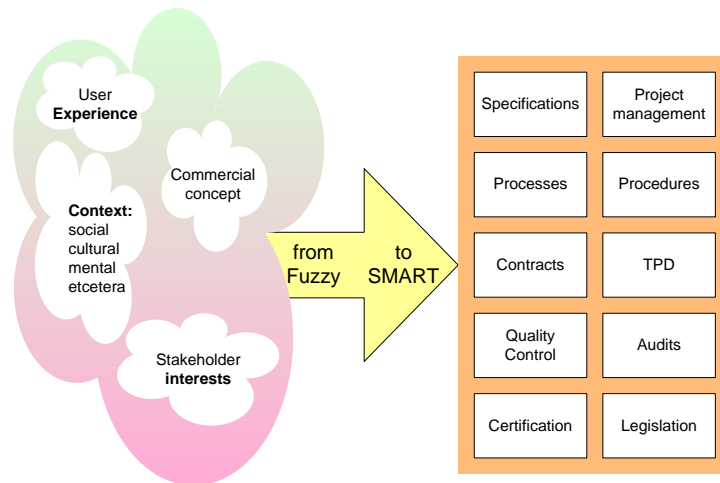


Figure 5.6: Problem (2): From Imagination to Formalization

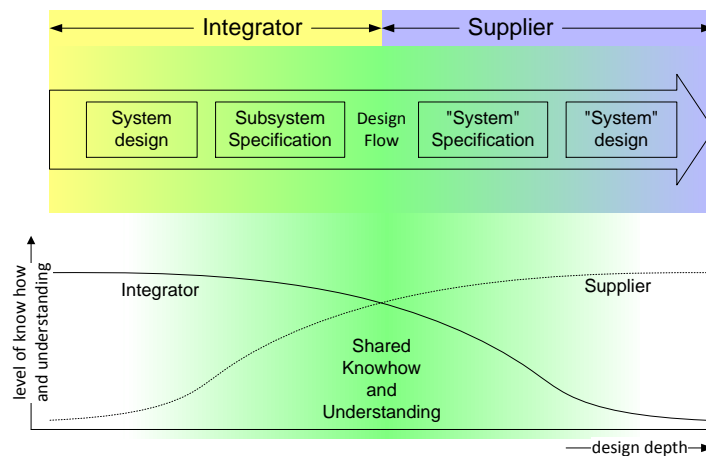


Figure 5.7: Theory: Subcontractors require SMART relation

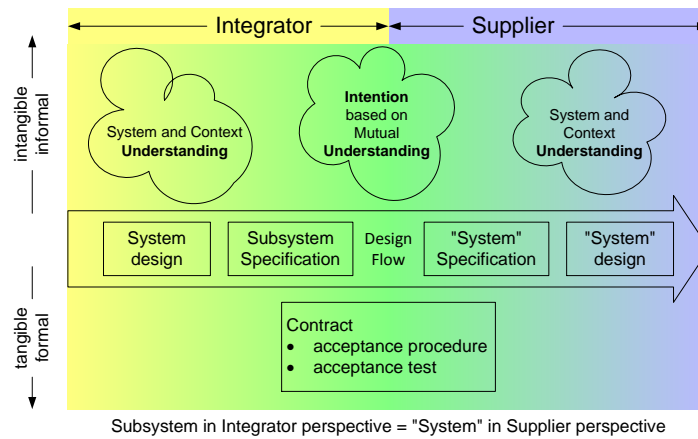


Figure 5.8: Critical Success Factor: Mutual understanding

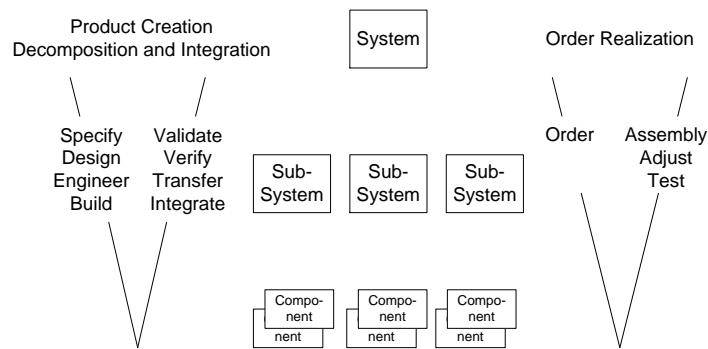


Figure 5.9: Views on Aggregation; Why SMART is needed

## 5.4 Examples of smartening fuzzy requirements

The user or consumer needs are shown in figure 5.10. As shown in this figure none of these requirements is specific nor measurable et cetera. However the qualified needs do provide insight in the motivation of users and in that way are useful to help others to obtain a better understanding.

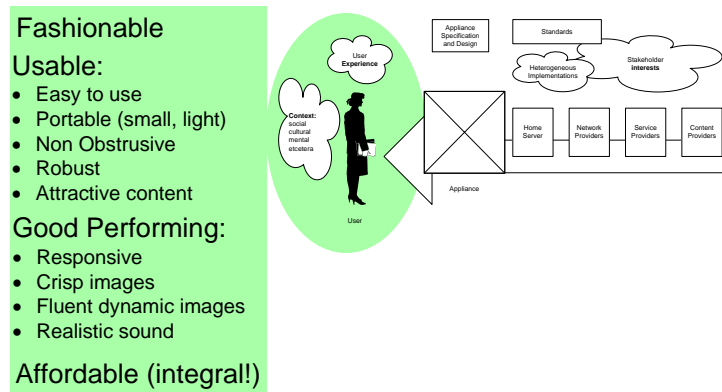


Figure 5.10: The "Fuzzy" needs of the User

Figure 5.11 shows in the same way the fuzzy needs of the providers and the retailers, which are also stakeholders of these products.

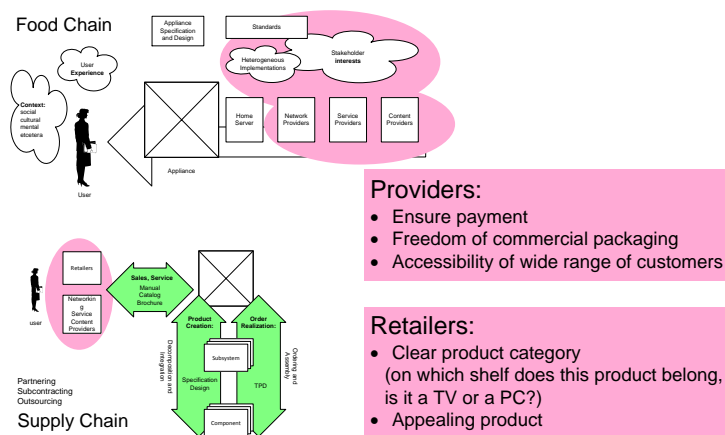


Figure 5.11: The "Fuzzy" needs of the Provider

The world of the designers is much less fuzzy. Systems are decomposed and interfaces are defined in SMART terms. Figure 5.12 shows the decomposition for this kind of product.

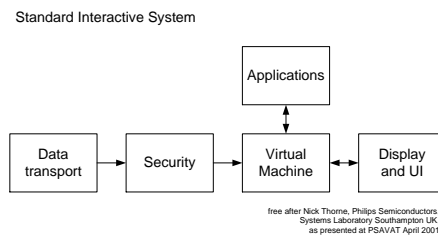


Figure 5.12: The "SMART" world of the Design

In such a decomposition many specification items can be defined. Figure 5.13 shows the decomposition annotated with specification items. Also a typical flow is shown for some user interaction: some request enters via the user interface, flows through all the blocks to the relevant service, and the answer travels in the opposite way. As an example we will have a look at the performance in terms of the response time.

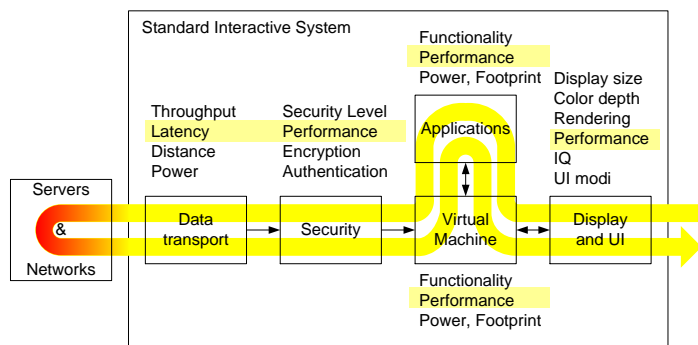


Figure 5.13: Specifiable characteristics

All functions in the chain contribute to the response time, figure 5.14 shows an (academic) budget for this response time. Note that the top part of the budget is well defined and in control of the appliance designer, however the lower levels of the budget are assumptions made by the appliance designer about the context. The wider the context becomes the more uncertainty will be present in the numbers.

The designer of the appliance need to make assumptions about the context in order to make a good design. These assumptions will be based on a model of the context. Such a model can be calibrated by measuring the context for as far as it exists already. However the designer should stay aware (as with all of his models) that this model is a tremendous simplification of reality. Reality is infinitely complex due to the possible variations in the food chain and the dynamics (changes over time).

times in milliseconds	Message Latency	Response Time
<b>Appliance</b>	<b>40</b>	<b>100</b>
Data transport	10	20
Security	10	20
Virtual Machine	10	20
Application	10	30
Graphics and UI	0	10
<b>Home Network</b>	<b>20</b>	<b>50</b>
Home Server	10	30
Network contention	10	20
<b>Provider Infrastructure</b>	<b>50</b>	<b>160</b>
Last-Mile network	10	20
Backbone network	20	40
Service server	10	50
Content server	10	50
<b>Total</b>	<b>110</b>	<b>310</b>
<b>User need</b>		<b>200</b>

All numbers are imaginary and for illustration purposes only

Figure 5.14: Response Time: Latency Budget

One of the important characteristics for the user of the appliance is the response time. As shown in figure 5.15 the model indicates good response times for functionality which stays within the appliance, but poor response times for functions which need interaction with far away servers. This insight will influence a lot of specification and design decisions. For all functions which require true interactive responses (i.e. less than 200 ms), some local solution is required, maybe supported by all kind of clever tricks such as look ahead, caching et cetera.

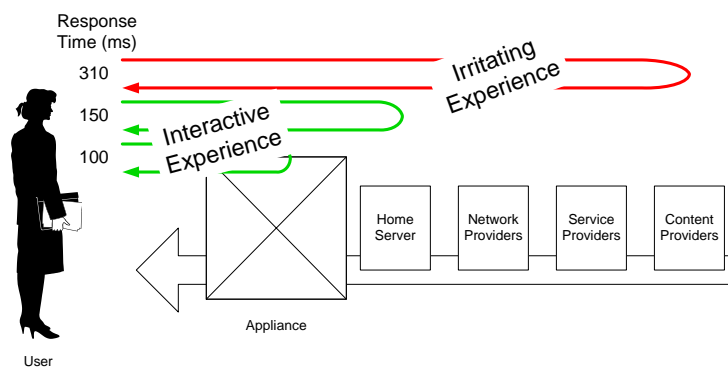


Figure 5.15: Interaction or Irritation?

A different area of fuzzy needs is image quality. The user needs good (sharp, bright, smooth moving) image quality. The verifiable image quality is often based

on synthetic images, which enable verification for all technical parameters. However a perfect technical image quality does not mean a satisfied user...

Some of the image quality aspects are even more fuzzy, for instance when perception is taken into account (color blind people!), or worse when taste comes into play (this image is too sharp, while someone else finds it too smooth).

Figure 5.16 visualizes these different types of needs.

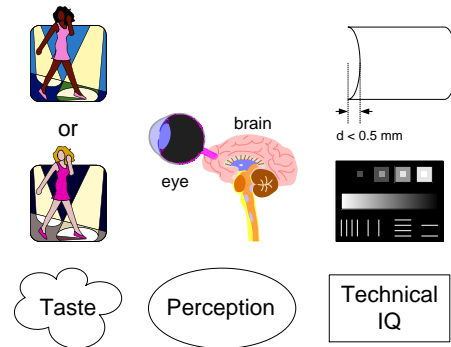


Figure 5.16: Image Quality

Fashion is also an intangible need of the user. However some product functions can be created to make it possible to follow the fashion, for instance by enabling personalization. A well known example is the exchangeable front of GSM phones. Figure 5.17 shows downloadable themes as example, which requires all kinds of functions such as format, download, import, scale et cetera.

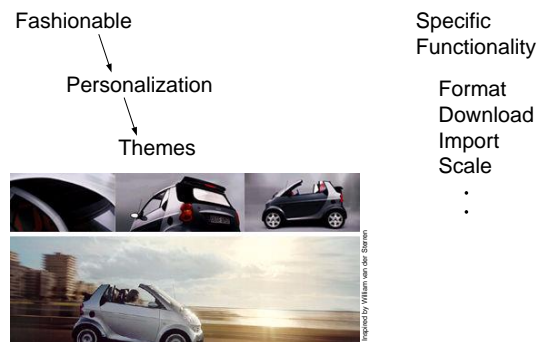


Figure 5.17: Fashionable

## 5.5 How to verify?

Verification of fuzzy requirements is difficult, while SMART requirements are verifiable by definition. However the fact that a smartened requirements is fulfilled, does not mean that the originating fuzzy requirement is also met.

Confrontation with market and consumers:

<b>Good</b>	<b>Bad</b>
Enthusiasm	Critical
Instant playing	Stumbling
Relaxed usage	Tension
Buying	Wait and see

Figure 5.18: From SMART to Fuzzy

Figure 5.18 shows that careful observation can be used to obtain insight in the level of fulfillment of the originating fuzzy requirement.

## 5.6 Conclusion

"Fuzzy" understanding of requirements and smartened descriptions of requirements are complementary. The smartening process of requirements often significantly increases the understanding of the requirements, mostly due to the need to articulate everything explicit. Unfortunately understanding itself is a non transferable concept, any description always flattens the rich understanding into a limited set of words and definition. Only readers with sufficient a priori knowhow are able to reconstruct the richer understanding again. The essential insights obtained in the requirements analysis are often captured in a few non-SMART statements, where the author hopes to enable the readers to obtain the original understanding.

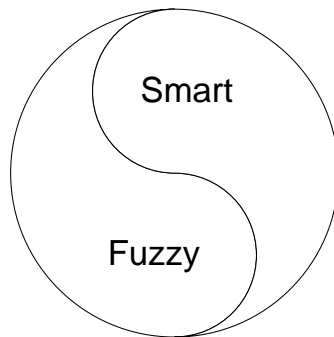


Figure 5.19: Complementing views

## 5.7 Acknowledgements

A long time ago Dieter Hammer send me a presentation explaining the SMART acronym. A vivid discussion with Thomas Gilb triggered this presentation, because I kept struggling with the need for being specific and measurable at the one hand, and the need to understand the less tangible aspects at the other hand. I thank them for their inspiration.

Tom Gilb and Lindsey Brodie helped me to trace back the origin of SMART.

## Chapter 6

# Communicating via CAFCR; illustrated by security example



### 6.1 Introduction

The communication aspect of architecting is discussed by means of an example product: mobile infotainment, see figure 6.1. This product is much more than only the tangible appliance: portable infotainment device, a long food chain to connect the appliance with the outside world is needed. The product can be used to watch movies or other content anywhere anytime, or to browse and update a calender and many more applications.

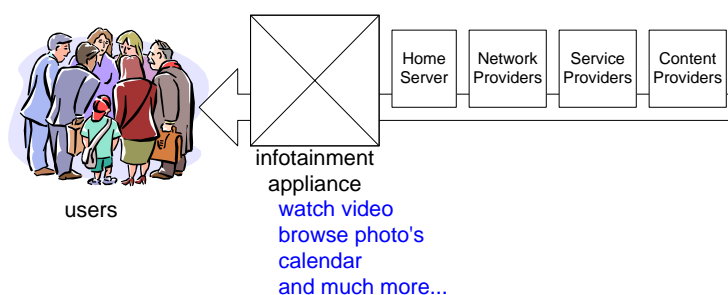


Figure 6.1: Example product: mobile infotainment

To make the example even more specific, the focus will be on security aspects. Of course many more aspects are important for this type of product, but security is

especially interesting for communication due to the wide range of (often conflicting) concerns with respect to security.

## 6.2 Stakeholders

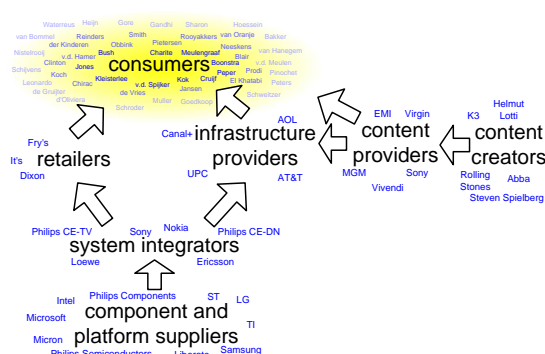


Figure 6.2: Value chain

The producer of the appliance for mobile infotainment is part of a much larger value chain, see figure 6.2. The food chain starts at the suppliers of components and platforms, such as Philips Semiconductors, Intel, Symbian and many more. These components are integrated by the appliance makers, such as Philips Consumer Electronics, Sony, Nokia or Samsung. Via a distribution chain of retailers and providers the appliance is delivered to a wide variety of consumers.

Complementary to this part of the value chain are an infrastructure value chain and content value chain. All kinds of players in these chains are mutually dependent: without content no appliance, but also without appliances no content.

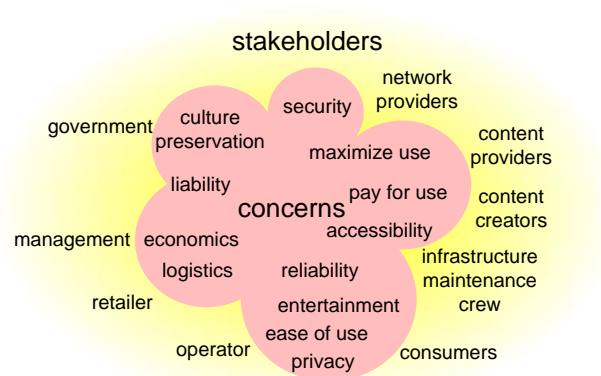


Figure 6.3: Stakeholders and concerns

Many stakeholders are involved in the creation of mobile infotainment. All of these stakeholders have multiple concerns, see figure 6.3. Although they use the

same label for a given concern, every stakeholder has its own specific interest and view on such a concern.

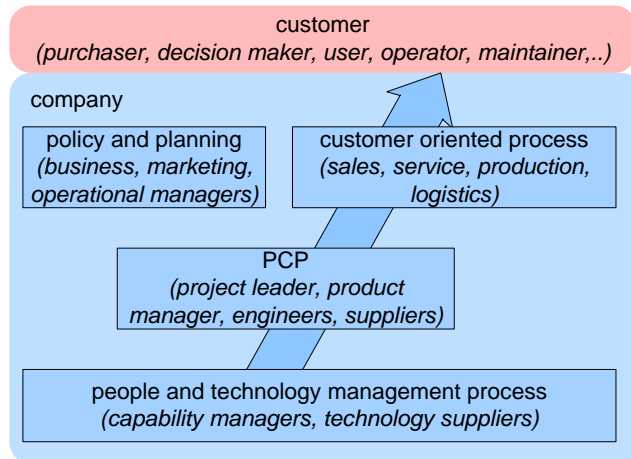


Figure 6.4: Internal stakeholders

Figure 6.3 shows predominantly the external stakeholders, but many (company-) internal stakeholders are involved as well, as modeled in figure 6.4. The internal stakeholders are supportive for the overall business goal, their organization to support such a new product is part of the creation of a new product.

## 6.3 The "CAFCR" model and qualities

A useful top level decomposition of an architecture is provided by the so-called "CAFCR" model, as shown in figure 6.5. 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!).

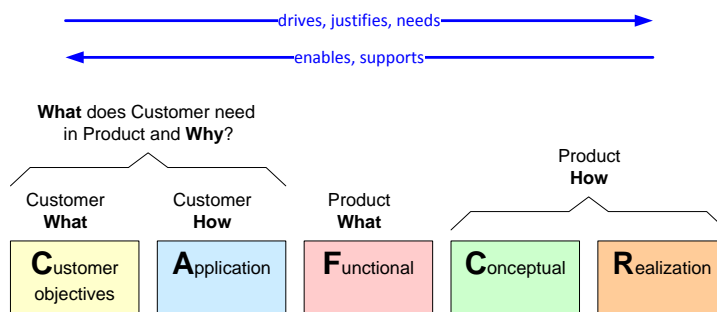


Figure 6.5: The "CAFCR" model

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 7.5.

In other words the views must be used concurrently, not top down like the waterfall model. However in 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 is used to provide a next level of reference models and methods [12]. 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.

"The customer" is a tremendous abstraction. Many players are involved in the value chain, while in many cases a player is a small company, where multiple people are involved. Figure 6.7 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.

The 5 CAFCR views become more useful when the information in one view is

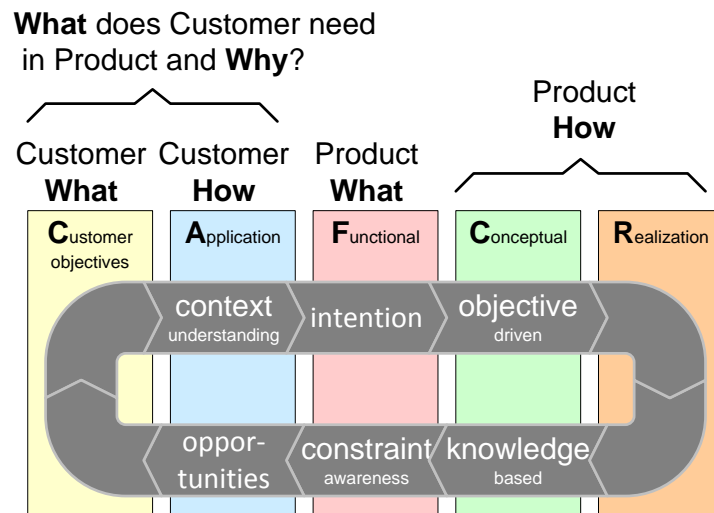


Figure 6.6: Five viewpoints for an architecture. The task of the architect is to integrate all these viewpoints, in order to get a *valuable, usable and feasible* product.

used in relation with neighboring views. One of the starting points is the use of the stakeholder concerns. Many stakeholder concerns are abstracted in a large set of more generic qualities. These qualities are meaningful in every view in their own way. Figure 6.8 shows the qualities as cross cutting needles through the CAFCR views.

## 6.4 Zooming in on security

As an example figure 6.9 shows security issues for all the views. The green (upper) issues are the desired characteristics, specifications and mechanisms. The red issues are the threats with respect to security. An excellent illustration of the security example can be found in [5].

### *Customer objectives view*

One of the typical customer objective with respect to security is to keep sensitive information secure, in other words only a limited set of trusted people has access. The other people (non trusted) should not be able to see (or worse to alter) this information.

### *Application view*

The customer will perform many activities to obtain security: from selecting trustful people to appointing special guards and administrators who deploy a security policy. Such a policy will involve classification of people with respect to need of

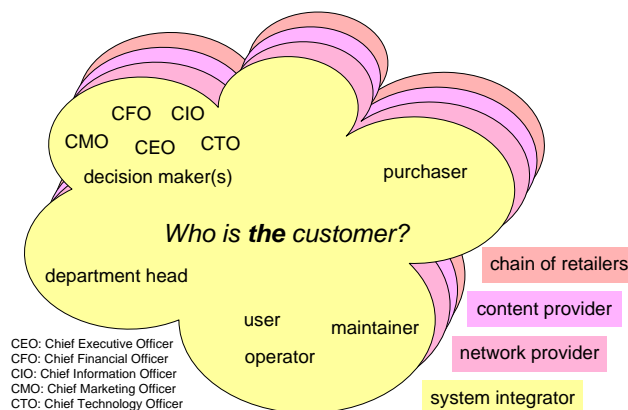


Figure 6.7: The abstracted customer

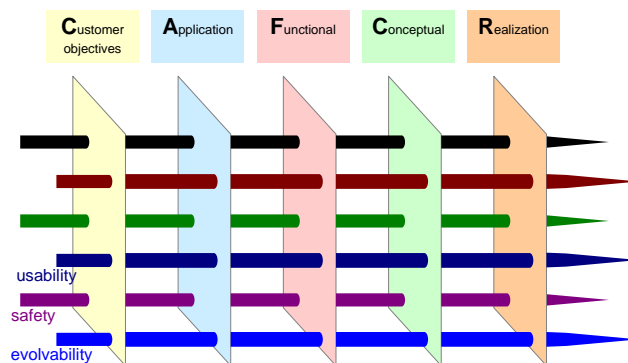


Figure 6.8: The quality needles are generic integrating concepts through the 5 CAFCR views

information and trustfulness and classification of information with respect to the level of security. To recognize *trusted* people authentication is required by means of badges, passwords and in the future additional biometrics. Physical security by means of buildings, gates, locks et cetera is also part of the customers security policy.

The security is threatened in many ways, from burglary to fraud, but also from simple issues like people forgetting their password and writing it on a yellow sticker. Social contacts of trusted people can unwillingly expose sensitive information, for instance two managers discussing business in a business lounge, while the competition is listening at the next table.

A frequent threat for security is formed by unworkable procedures. For instance the forced change of passwords every month, resulting in many people writing

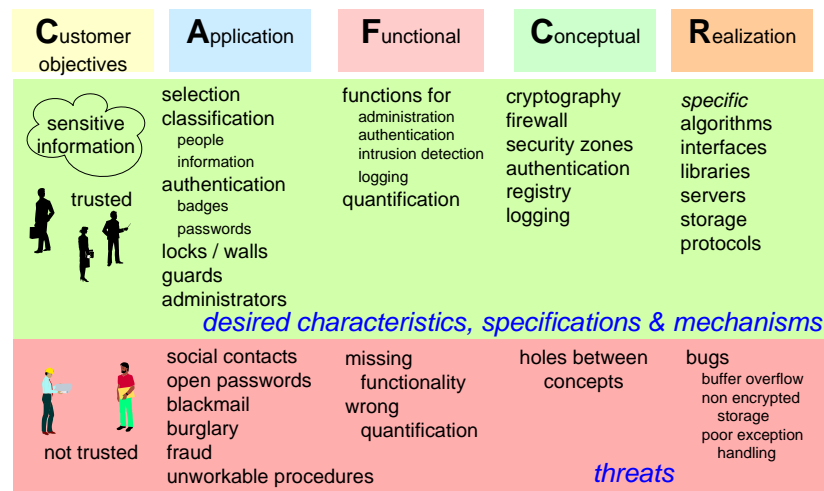


Figure 6.9: Example security through all views

down the password.

An interesting article is [2], which shows how secret security procedures, in this case for passenger screening at airports, is more vulnerable. It describes a method for terrorists how to reverse engineer the procedures empirically, which turns the effectiveness of the system from valuable to dangerous.

#### *Functional view*

The system under consideration will have to fit in the customers security. Functions for authentication and administration are required. The performance of the system needs to be expressed explicitly, for instance the required confidence level of encryption or the speed of authentication.

Security threats are mostly caused by missing functionality or wrong quantification. This threat will surface in the actual use, where the users will find work around compromising the security with the work around.

#### *Conceptual view*

Many technological concepts have been invented to make systems secure, for example cryptography, firewalls, security zones, authentication, registry, and logging. Every concept covers a limited set of aspects of security. For instance cryptography makes stored or transmitted data non-interpretable for non trusted people.

Problems in the conceptual view are mostly due to the non ideal combination of concepts. For instance cryptography requires keys. Authentication is used to access and validate keys. The interface between cryptography and authentication is a risky issue. Another risky issue is the transfer of keys. All interfaces between the concepts are suspicious areas, where poor design easily threatens the security.

#### *Realization view*

The concepts are realized in hardware and software with specific algorithms, interfaces in specific libraries, running at specific clients and servers et cetera. Every specific hardware and software element involved in the security concepts in itself must be secure, in order to have a secure system.

A secure realization is far from trivial. Nearly all systems have bugs. Well known security related bugs are buffer overflow bugs, which are exploited by hackers to gain access. Another example is storage of very critical security data, such as passwords and encryption keys, in non encrypted form. In general exception handling is a source of security threats in security.

#### *Security conclusion*

Security is a quality which is heavily determined by the customers way of working (application view). To enable a security policy of the customer a well designed and implemented system is required with security functionality fitting in this policy.

In practice the security policy of customers is a large source of problems. Heavy security features in the system will never solve such a shortcoming. Another common source of security problems is poor design and implementation, causing a fair policy to be corrupted by the non secure system.

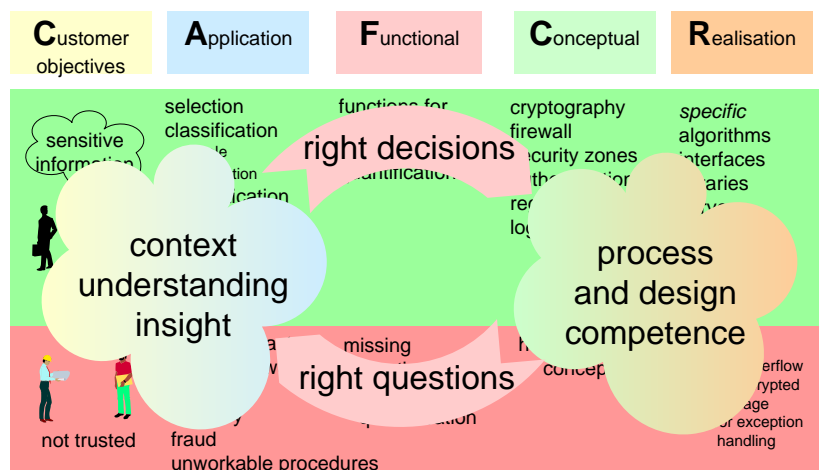


Figure 6.10: Role of views

Figure 6.10 visualize the reasoning with respect to security over the different views. Only if sufficient understanding of the context is combined with good process and design competences an acceptable result can be obtained.

Note that a very much simplified view on security is presented, with the main purpose of illustration. A real security view will be more extensive than described here.

## 6.5 The wonder of communication

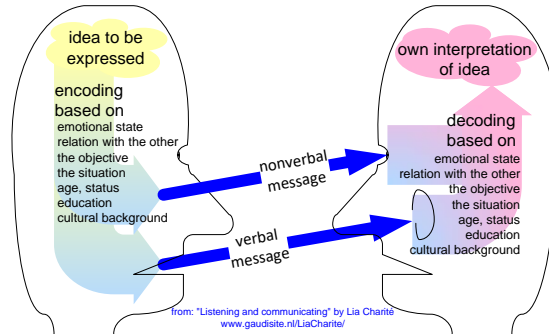


Figure 6.11: Active listening: the art of the receiver to decode the message

If someone wants to transfer an idea to another person, then this idea is encoded in a message. This message is encoded by a variety of means, ranging from the verbal message to the non verbal message such as facial expression(s), gestures and voice modulation. The encoding of this message depends on many personal aspects of the *speaker*, see figure 6.11. The receiver of this message has to decode this message and makes his own interpretation, also based on many personal aspects of the *receiver*.

From technical point of view a pure miracle is happening in communication: sender and receiver use entirely different configured encoders and decoders and nevertheless we are able to convey messages to others.

to calibrate:  
repeat many times with different  
examples, illustrations, and explanations

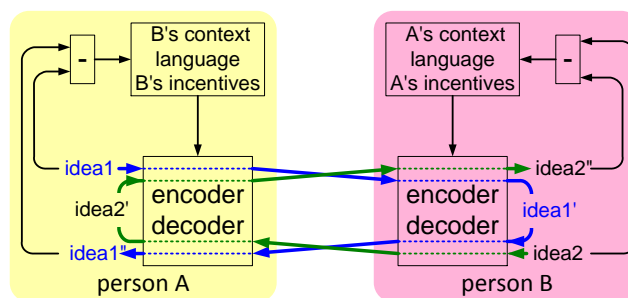


Figure 6.12: Intense interaction needed for mutual understanding

The mechanism behind this miracle can be understood by extending the model of sender and receiver as in figure 6.12. The mutual understanding is built up in an

interactive calibration process. By phrasing and rephrasing examples, illustrations and explanations the coding and decoding information is calibrated.

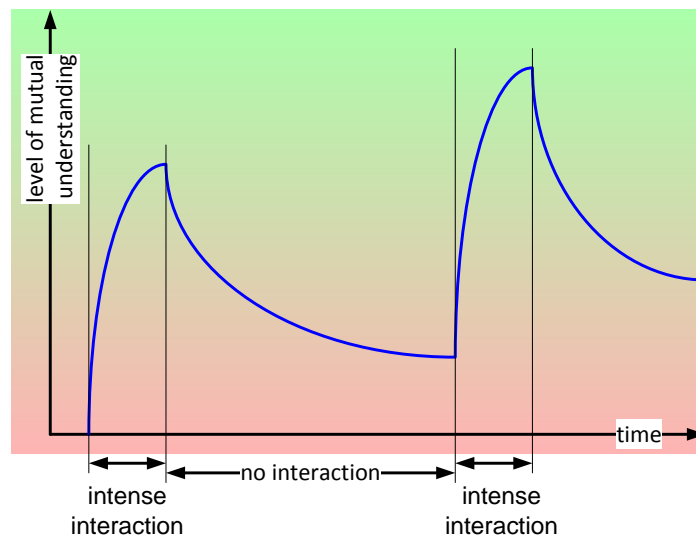


Figure 6.13: Mutual understanding as function of time

The calibration information is very dynamic, part of the coding depends on volatile issues, such as mood, and context. During interaction the mutual understanding improves, while it degrades as long as no interaction takes place, as visualized in figure 6.13.

Note that glossaries of terms, unified notations and all these kind of measures do not fundamentally address the communication difficulties explained here. In fact standardized terminology and notations are minor<sup>1</sup> in comparison with the human differences which have to be bridged continuously.

<sup>1</sup>Dogmatic applied unification of terms and notations work in my experience often counterproductive. Problems or viewpoints which are more easily expressed in other terms are disallowed due to the unification obsession, where active participation is required to obtain understanding that exceeds terms and notations.

## 6.6 Story telling

Story telling is a method to enable communication between people with different points of view. The method is a means to get discussions quickly a concrete and factual.

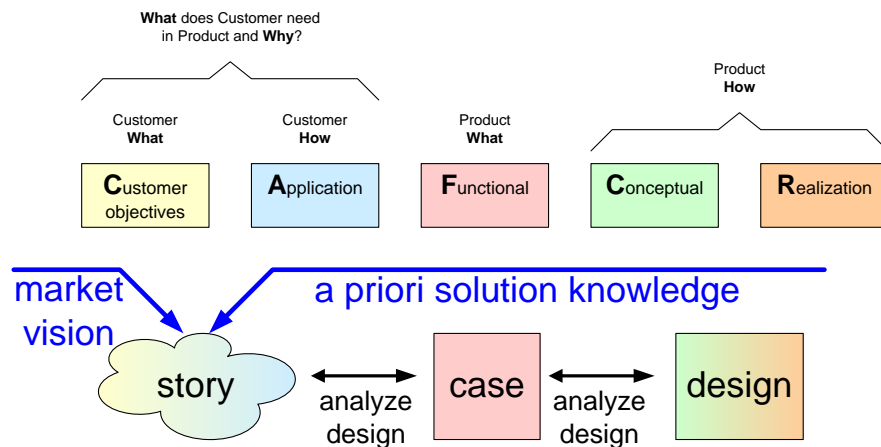


Figure 6.14: Story telling method

Figure 6.14 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.

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.

The strength of the method is early focus on concrete actual problems and solutions. Once sufficient factual specification and design depth is obtained, it becomes time to determine useful generic concepts.

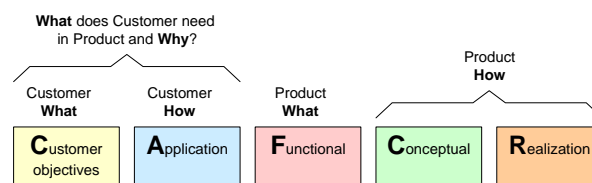
## 6.7 Summary

The previous chapters have shown that many stakeholders with many different concerns are involved. Also is shown how difficult a bilateral communication is. The challenge of developing a complex product, such as the mobile infotainment, is to communicate with many different stakeholders over many different subjects. Figure 6.15 summarizes this by showing a small subset of stakeholders, one of their most primary thoughts and the bad consequences if this thought is followed without taking other concerns into account.

stakeholder	primary thought	threat
consumer	privacy	kill usability
content provider	DRM, consumer == pirate	kill usability kill market
Chief Financial Officer	how to stay in control	kill usability
operational manager	result in time, accessibility	security
web engineer	PHP only supports alphanumerical password	poor password protection
crypto engineer	128 bit keys	no attention for key handling process

Figure 6.15: How do these stakeholders communicate?

Figure 6.16 summarizes the contribution of the "CAFCR" model in the communication.



CAFCR, as shared reference, enables:

- + Positioning of concerns, problems and solutions
- + Checklists per view
- + Reasoning top down and bottom up

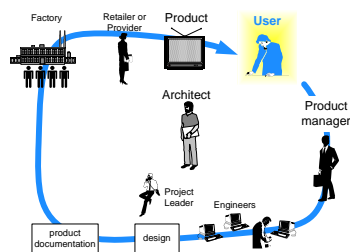
Figure 6.16: Summary

## 6.8 Acknowledgements

Henk Koning helped by providing inputs in the initial setup of the presentation. The definition of active listening and the diagram of bilateral communication are reused from a presentation given by my wife Lia Muller for her psycho-social study. Peter van den Hamer proposed several textual improvements.

## Chapter 7

# Architect and Human Measure; the integration role



### 7.1 Introduction

This article is written as part of a collective effort to write a book about "ICT and the Human Measure". It will become one chapter of this book, describing "the role of the architect". For that reason the problem statement is short and illustrative only.

### 7.2 Illustration of the problem




- A ☐  depressed
- B ☐  desperate
- C ☐  hysteric

Figure 7.1: Did you ever program a VCR?

Many products have characteristics which are determined by technology push rather than user need. Take for instance most video recorders, which are often way too difficult to program for ordinary (non-technical) people. This is illustrated by figure 7.1.

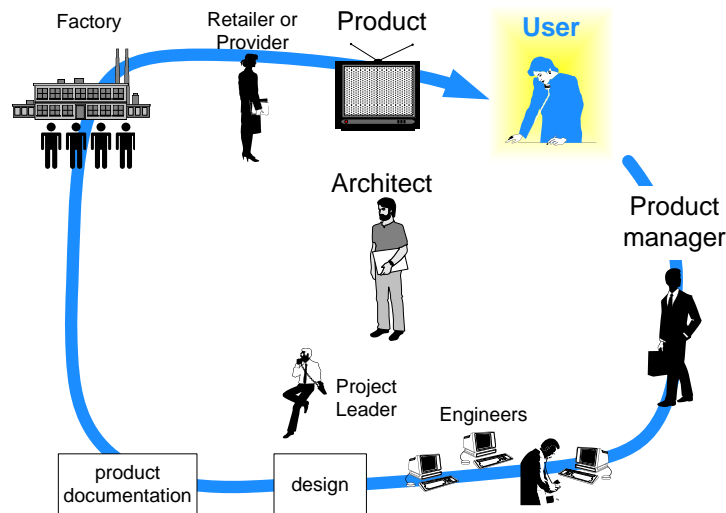


Figure 7.2: Product Creation Cycle

One cause of this problem is the long chain of activities which results in a product, as shown in figure 7.2. This long chain of activities also involves many different stakeholders, ranging from potential customers, and product managers to development engineers and production personnel.

A lot of the stakeholders "live" in the engineering world, which addresses a lot technology concerns. The other end of the stakeholder chain, the human users, live in the real world, with many human concerns, such as emotions, feelings, perceptions et cetera. Figure 7.3 visualizes the gap between those stakeholders.

Both figures 7.2 and 7.3 already hint at the crucial role played by the architect by architecting the solution.

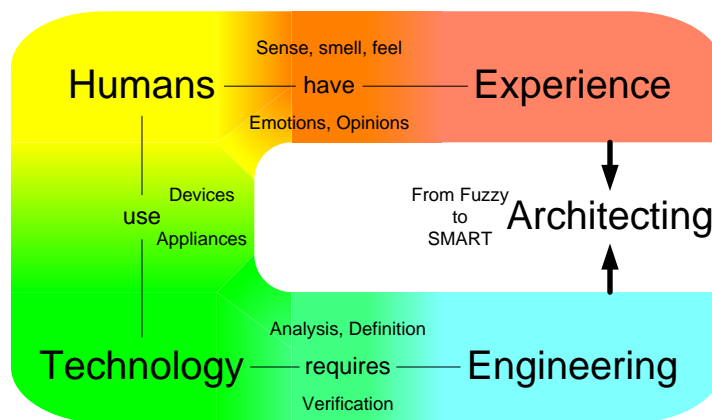


Figure 7.3: Bridging the gap between Human Experience and Engineering

## 7.3 What is architecting?

Architecting in product creation spans from *understanding* the **why**, via *describing* the **what** to *guiding* the **how**, as shown in figure 7.4. Or in even more popular terms: *do the right things* and *do the things right*

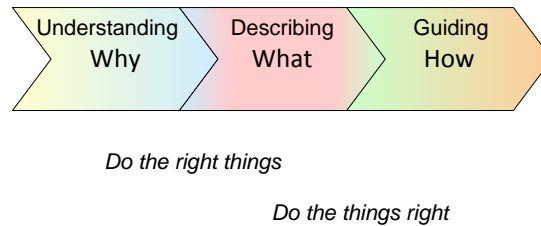


Figure 7.4: Architecting visualized

Architecting is a job which is done by all members of the product creation team, however the architect is responsible for the consistency and balance of **why**, **what** and **how**

A useful top level decomposition of an architecture is provided by the so-called "CAFCR" model, as shown in figure 7.5. 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!).

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).

The **how** of the product is created by many specialists. The **how** is guided by the architecture. At least 5 views are required for guidance:

- functional decomposition
- construction decomposition
- allocation of functions to construction elements
- infrastructure
- integrating concepts

Figure 7.6 visualizes these 5 **how** views.

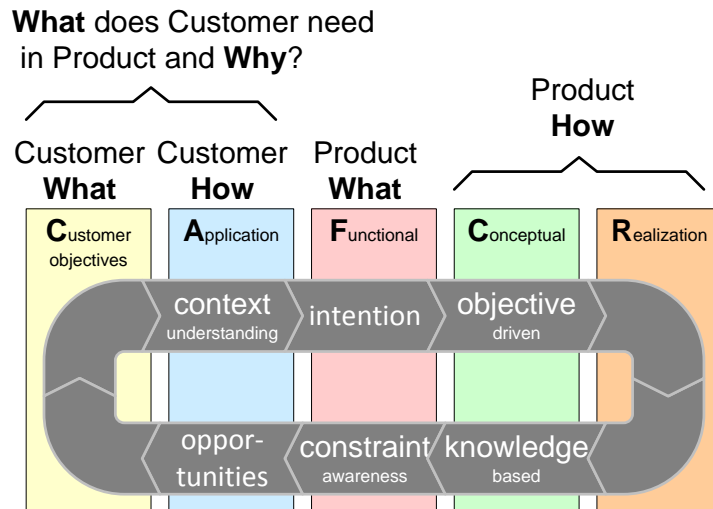


Figure 7.5: Five viewpoints for an architecture. The task of the architect is to integrate all these viewpoints, in order to get a *valuable, usable and feasible* product.

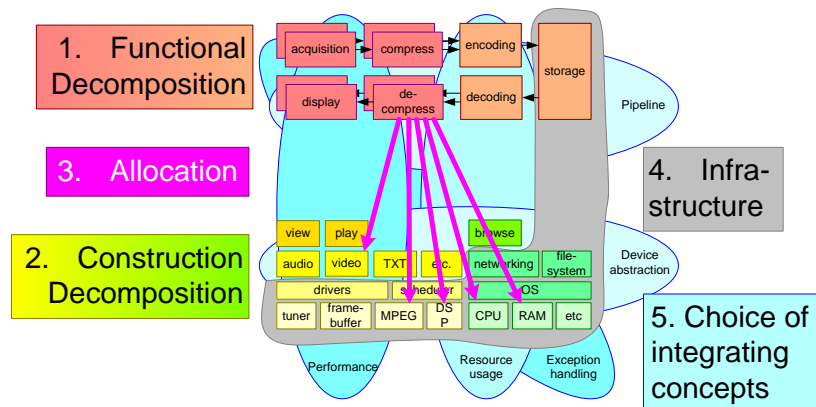


Figure 7.6: Guiding **how** by providing five *how*-viewpoints

## 7.4 The architect

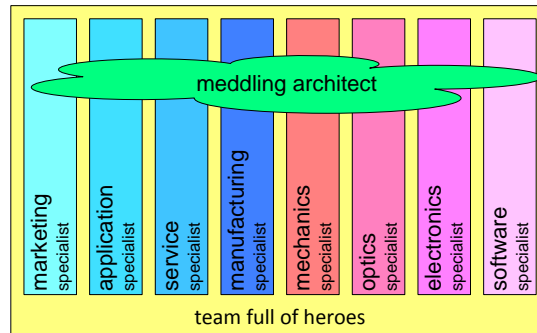


Figure 7.7: The architect integrating all specialist teamplayers

The architecting function is ideally performed by every teammember. However the primary responsibility for the balance and consistency of requirements, specification and design is owned by the architect. Figure 7.7 shows the architect meddling with the work performed by all teammembers, in order to obtain the balance and maintain the consistency.

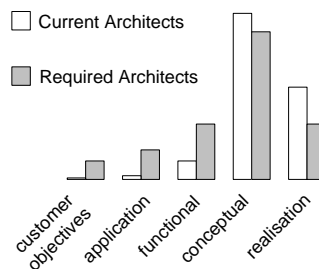


Figure 7.8: Required architect know-how per view, typical for current architects and the preferred profile.

The role of the architect is to (proactively) integrate the work of all the specialists. This role requires sufficient know-how in the five views, see figure 7.8. Most current architects have a dominant technical view on the world and should acquire more know-how from the customer world.

This broad profile of the architect does not evolve automatically. Potential architects grow by stepwise broadening their scope, see figure 7.9. The intermediate roles are quite important in complex systems, it prevents the need for an impossible broad and heavy superarchitect.

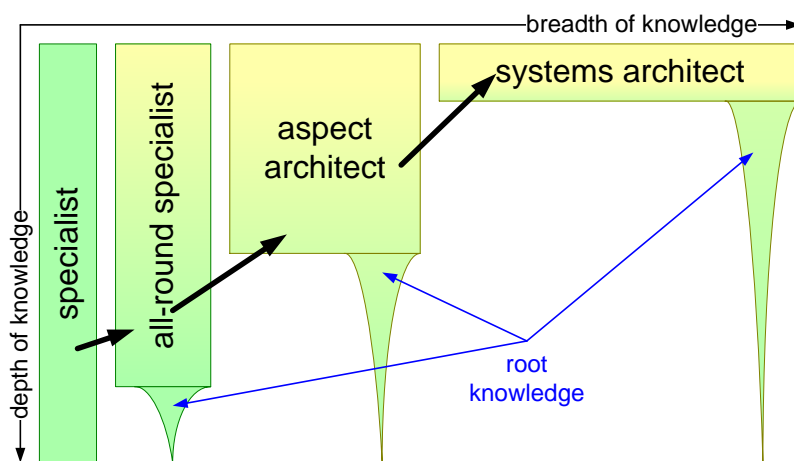


Figure 7.9: The architect maintains technical roots

## Chapter 8

# Architecture; the building as a product



### 8.1 Introduction

The formal opening of the new IST<sup>1</sup> building of Philips Research provided an opportunity to compare architecting buildings and architecting electronic products. Many IST people are involved in architecting electronic products.

### 8.2 The Product: Building WDC

Figure 8.1 shows the new IST building, which is the product of a significant architecting effort.

Architecting a product involves 3 major phases, see also figure 8.2:

- Understanding **why**
- Defining **what**
- Guiding **how**

---

<sup>1</sup>Information and Software Technology



Figure 8.1: The product

The **why** and the **what** of a product are directly derived from the needs and the constraints of the stakeholders. The **what** is consolidated in a requirement specification. To determine the **what** sufficient understanding of the **how** is needed to ensure the feasibility. Understanding the **how** requires a significant amount of technology and construction know-how.

The authentic objectives of the Philips management with respect to the Campus can be read (in dutch) in figure 8.3. These objectives and subgoals are translated in English, see tables 8.1 and 8.2.

- Stimulating working environment for synergy and innovation
- Enduring development of the organization
- Efficient accomodation

Table 8.1: *Objectives of the Philips management w.r.t. the Campus*

The objectives make it clear that the Philips management is aware of influence of the housing on the organization and may other human factors. The objectives address mostly human factors, within an economic efficiency constraint.

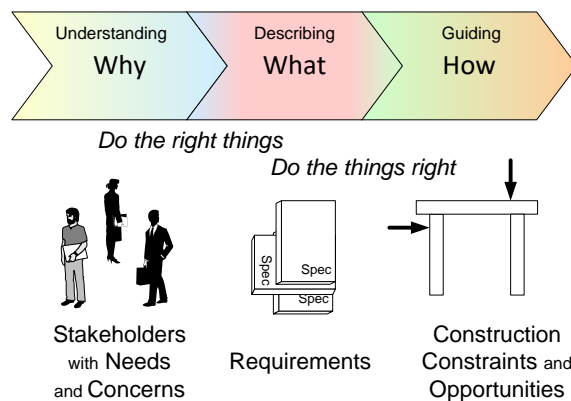


Figure 8.2: What is Architecting?

#### Campus Doelstellingen

- \* Stimulerende werkomgeving voor synergie en innovatie
- \* Duurzame ontwikkeling van de organisatie
- \* Efficiënte huisvesting

#### Sub-doelen :

- + Open relatie met omgeving
- + Innovatieve werkomgeving
- + Bevorderen van synergie d.m.v. gemeenschappelijke faciliteiten
- + Integratie van werken en privé
- + Blijvende positie van Philips onder de eerste elektronica concerns
- + Versnelling van innovatie-processen
- + Aantrekkingskracht toptalent
- + Opheffen versnippering
- + Versterken imago

Figure 8.3: Philips management objectives w.r.t. Campus

The vision of the architects, see figure 8.4 is to create a very *open* and *transparent* building. *Open* and *transparent* stimulates *communication*, *sharing* and *cooperation*. An modern outlook and embedding the building in the high-tech campus creates a stimulating and innovative environment.

Somewhat late in the process the final inhabitants were involved in the internal design of the building. Table 8.3 show some of the wishes of the inhabitants. especially the *concentration* requirement was undervalued by the architects and some raging debates took place about this subject.

Figure 8.5 shows the internal design after taking into account the *concentration* requirement. The open space is mostly filled with (small) two person rooms, to provide the required quiet atmosphere. The informal communication is now foreseen near the two coffee pantries.

The openness is reduced to open staircases and the atrium at the south side of

- Open relation with environment
- Innovative working environment
- Encouraging synergy by sharing facilities
- Integration of professional and private life
- Consolidation of position of Philips as one of the leading electronic companies
- Acceleration of innovative processes
- Remove fragmentation
- Improve image

Table 8.2: *Subgoals w.r.t. the Campus*

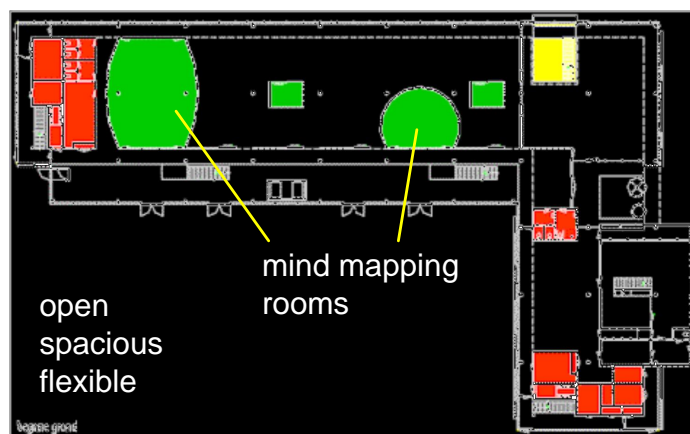


Figure 8.4: The architects vision

the building. Figure 8.6 shows an impression of the space near the staircase.

The work of the architect is to bridge the stakeholder world and the construction world. This construction world is much more technical, many technical aspects must be taken into account by the architect.

Figure 8.7 shows a number of the more technical aspects:

- Facilities
- Infrastructure
- Design aspects
- Construction aspects

Note that most technical aspects have a counterpart in terms of stakeholder concerns. For instance safety is a major concern of a plant manager, which results

- Comfortable
- Concentration
- Communication
- Practical

Table 8.3: *Wishes and concerns of the inhabitants*

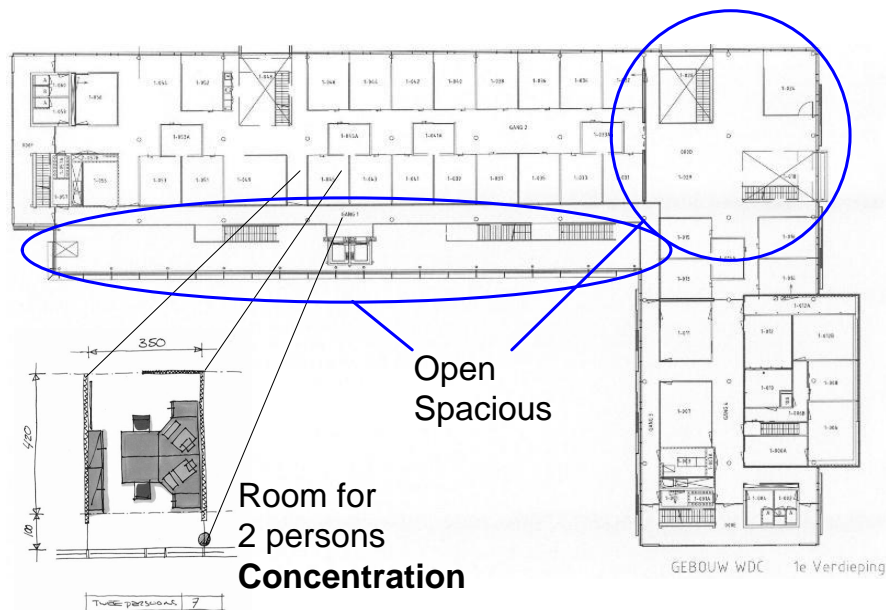


Figure 8.5: After user amendment

in many procedures, guidelines, provisions and hence also in requirements for the new building.

In technical architecting we promote the "CAFCR"-model[13]. This model provides a spectrum of 5 viewpoints, ranging from the customer objectives (customer **what**) to the realization (product **how**). Figure 8.8 shows the architecting aspects mapped on this model.



Figure 8.6: Space impression

<b>infrastructure</b>	<b>design aspects</b>
+ power	+ maintenance
+ telecom and computer network	+ safety
+ climate control	+ security
+ light	+ flexibility
+ fire detection and prevention	+ campus style
<b>facilities</b>	<b>construction aspects</b>
+ sanitary	+ legislation
+ catering	+ material properties
+ meeting	+ weight, size, strength
	+ cost, effort
	+ tools

Figure 8.7: The technical side of the architecture

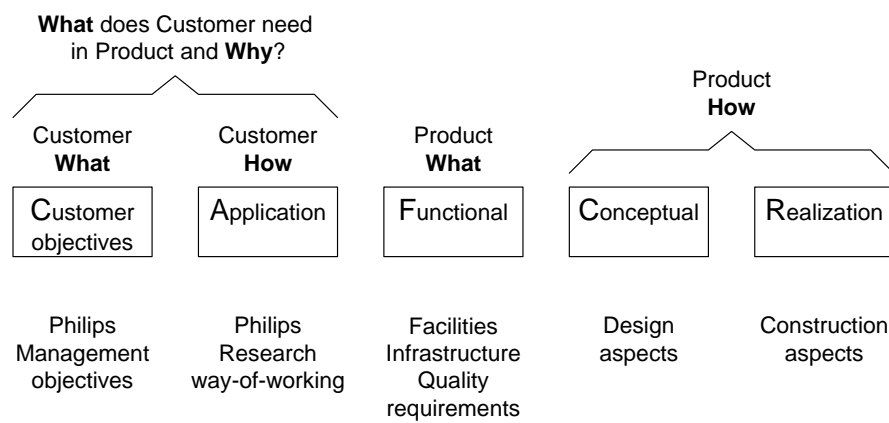


Figure 8.8: WDC architecting mapped on "CAFCR"

### 8.3 The Product: Digital Video Recorder

A Digital Video Recorder (DVR) is described in the same perspective as the building WDC to compare building architecting and electronic product architecting. One of the main functions of a DVR is *time shifting*, providing a consumer freedom in time. Figure 8.9 shows a simple example of the application of *time shifting*.

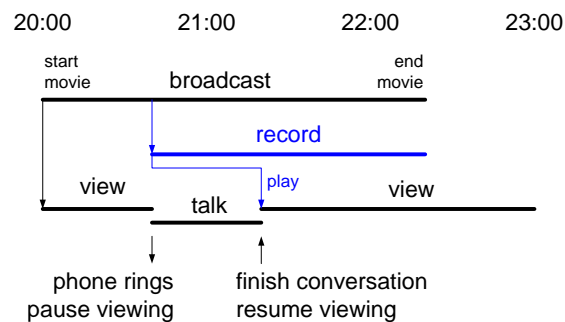


Figure 8.9: Example product: Digital Video Recorder

Some objectives of the consumer are summed up in table 8.4.

- **Time independent** entertainment and other video content
- Convenience, no hassle
- Fits in family environment

Table 8.4: *Consumer Objectives*

The product requirements are translated in a design in an iterative fashion. Figure 8.10 shows some of the relevant technical side of a DVR.

Finally all these issues are mapped on the "CAFCR"-model, see figure 8.11. Many similarities of the WDC and the DVR "CAFCR"-model are evident.

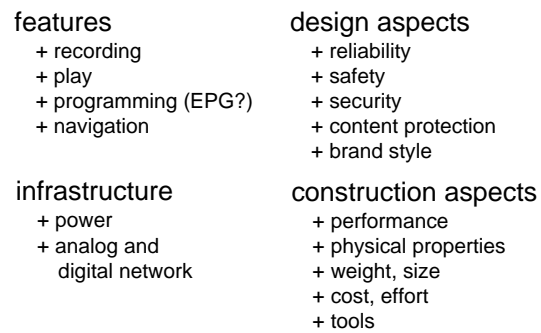


Figure 8.10: The technical side of the architecture

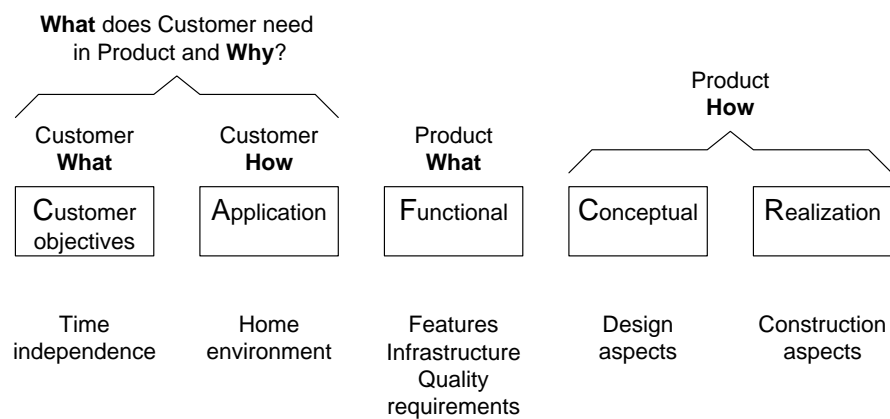


Figure 8.11: Product architecting mapped on "CAFCR"

## 8.4 Discussion

Many similarities exist between the building architect and the (electronics) product architect. The maturity of both disciplines is quite different. Architecting buildings is a very mature discipline, with millenia of experience. At the other hand electronics and software are very young disciplines, where the construction technology is changing rapidly. The (electronics) product architecting is a very immature discipline.

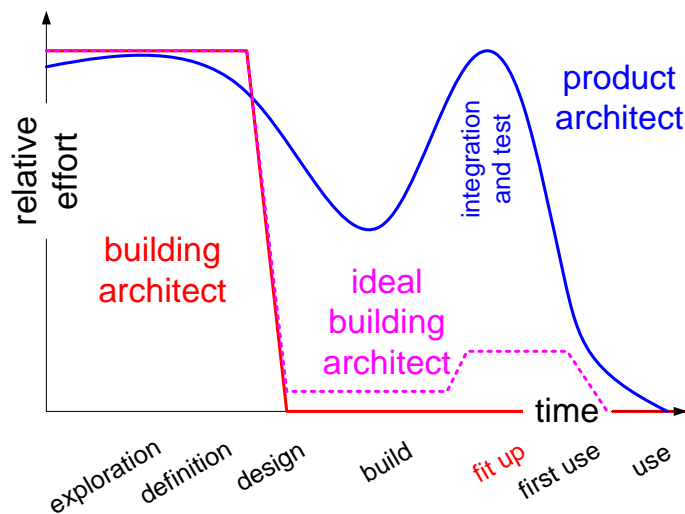


Figure 8.12: Architecting dynamics

The maturity influences the way of working of the architect. The building architect does most, if not all, work at the beginning of a project. Figure 8.12 shows the relative effort as function of time, showing that the building architect stops after delivering the design. The building architect is not involved in building, preparing the use, or the actual use of the building. The building architect is sometimes involved in the acceptance of the building, the phase transition between building and using.

The product architect must be involved in the later stages of the project, to solve unforeseen requirements and implementation hurdles. Due to the immaturity of the involved disciplines both unforeseen requirements and implementation hurdles are a fact of life. If the architect is not available in that phase the integrity of the architecture is in severe danger.

The strict phasing of the building world is in my opinion too strong. Many building architects don't get practical feedback from builders and users. Also many unforeseen requirements and implementation hurdles are also a fact of life

in buildings. As example of an unforeseen requirement we can look at the *concentration* wish of the users, researchers, which had a significant impact on the architecture. Examples of implementation hurdles are the climate control and the light controls.

## 8.5 Conclusion

The main function of an architect is to bridge the stakeholder world and the construction world. This requires substantial know how and understanding of both worlds. The function of the building architect and the electronic product architect is quite similar from that perspective, see figure 8.13.

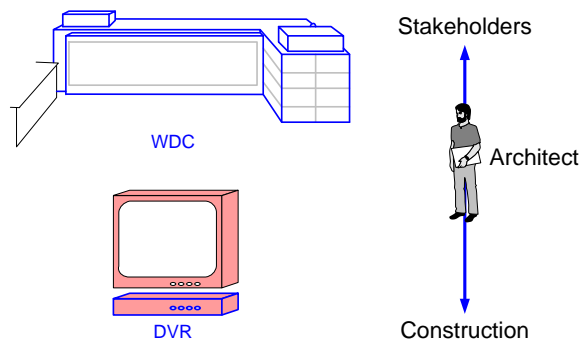


Figure 8.13: Bridging 2 worlds

# Bibliography

- [1] Frederick P. Brooks. *The Mythical Man-Month*. Addison Wesley, 1975, ca. 1995.
- [2] Samidh Chakrabarti and Aaron Strauss. Carnival booth: An algorithm for defeating the computer-assisted passenger screening system. <http://swissnet.ai.mit.edu/6805/student-papers/spring02-papers/caps.htm>, 2002. Shows that security systems based on secret designs are more vulnerable and less secure.
- [3] Hay Management Consultants. Technology management cycle. Hay Managament Consultants showed me this model in 1997/1998, taken from an article by a Japanese author. The original title of the Japanese article is unknown.
- [4] George T. Doran. There's a S.M.A.R.T. way to write management's goals and objectives. *Management Review (AMA Forum)*, pages 35–36, November 1981.
- [5] Charles C. Mann. Homeland insecurity. *The Atlantic Monthly*, pages 81–102, September 2002. Volume 290, No. 2T; Very nice interview with Bruce Schneier about security and the human factor.
- [6] Gerrit Muller. The arisal of the system architect. <http://www.gaudisite.nl/MaturityPaper.pdf>, 1999.
- [7] Gerrit Muller. The system architecture homepage. <http://www.gaudisite.nl/index.html>, 1999.
- [8] Gerrit Muller. Case study: Medical imaging; from toolbox to product to platform. <http://www.gaudisite.nl/MedicalImagingPaper.pdf>, 2000.
- [9] Gerrit Muller. Process decomposition of a business. <http://www.gaudisite.nl/ProcessDecompositionOfBusinessPaper.pdf>, 2000.

- [10] Gerrit Muller. From the soft and fuzzy context to SMART engineering. <http://www.gaudisite.nl/FromFuzzyToSmartPaper.pdf>, 2001.
- [11] Gerrit Muller. Function profiles; the sheep with 7 legs. <http://www.gaudisite.nl/FunctionProfilesPaper.pdf>, 2001.
- [12] Gerrit Muller. Architectural reasoning explained. <http://www.gaudisite.nl/ArchitecturalReasoningBook.pdf>, 2002.
- [13] 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.
- [14] Eberhardt Rechtin and Mark W. Maier. *The Art of Systems Architecting*. CRC Press, Boca Raton, Florida, 1997.

## History

**Version: 0.2, date: June 16, 2006 changed by: Gerrit Muller**

- Added "Decomposing the Architect; What are Critical Success Factors?"

**Version: 0.1, date: October 18, 2002 changed by: Gerrit Muller**

- Added "Communicating via CAFCR; illustrated by security example"

**Version: 0, date: June 12, 2002 changed by: Gerrit Muller**

- Created very preliminary bookstructure, no changelog yet