

Tutorial Human Side of Systems Architecting



Gerrit Muller

Buskerud University College and Embedded Systems Institute

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

`gerrit.muller@embeddedsystems.nl`

Abstract

Architects play a crucial role in creating systems that fit well in the human needs. The creation of these systems requires many human interactions between all stakeholders. The background of architects, however, is completely different, mostly technical. We bring insight in the human aspects of systems architecting and we provide an approach with related tools to address the human aspects.

Copyright © 2008 by Gerrit Muller. Published and used by INCOSE with permission.

Distribution

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

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

Contents

1	Introduction	1
2	Human Factors in Defense	2
2.1	Human Factors in Defense	2
2.2	Acknowledgements	3
3	The Role and Task of the System Architect	4
3.1	Introduction	4
3.2	Deliverables of the System Architect	4
3.3	System Architect Responsibilities	5
3.4	What does the System Architect do?	8
3.5	Task versus Role	9
3.6	Acknowledgements	9
4	The Awakening of a System Architect	11
4.1	Introduction	11
4.2	The Development of a System Architect	11
4.3	Generalist versus Specialist	12
4.4	Acknowledgements	14
5	Architecting for Humans; How to Transfer Experience?	15
5.1	Introduction	15
5.2	User Experience	18
5.3	Engineering	23
5.4	Education	25
5.5	Conclusion	27
5.6	Acknowledgements	28
6	Human Side: Interpersonal Skills	29
6.1	Introduction	29
6.2	The Wonder of Communication	30
6.3	Interpersonal skills	32

6.4	Acknowledgements	36
7	Human Side: Team Work	37
7.1	Why Work in Teams?	37
7.2	Team size	38
7.3	Team composition	39
7.4	The Process of Creating and Employing a Team	40
7.5	Housing and Location	42
7.6	Concurrency	43
7.7	Critical success factors	45
7.8	Acknowledgements	46
8	Architecting Interaction Styles	47
8.1	Introduction	47
8.2	Provocation	47
8.3	Facilitation	48
8.4	Leading	48
8.5	Empathic	49
8.6	Interviewing	49
8.7	White-board simulation	49
8.8	Judo tactics	50
9	Function Profiles; The Sheep with Seven Legs	51
9.1	Introduction	51
9.2	Systems Architect Profile	52
9.2.1	Most discriminating characteristics	52
9.3	Test Engineer Profile	53
9.4	Developer Profile	53
9.5	Operational Leader Profile	54
9.6	Line Manager Profile	54
9.7	Commercial Manager Profile	55
9.8	Definition of Characteristics	55
9.8.1	Interpersonal skills	55
9.8.2	Know-how	56
9.8.3	Reasoning Power	57
9.8.4	Executing Skills	57
9.8.5	Process Skills	58
9.8.6	Project Management Skills	58
9.8.7	Commercial Skills	59
9.8.8	Human Resource Management Skills	59
9.9	Acknowledgements	59

10 Short introduction to basic “CAFCR” model	60
10.1 Introduction	60
10.2 The CAFCR model	61
10.3 Who is the customer?	61
10.4 Life Cycle view	62
10.5 Zooming in on security	64
11 Story How To	67
11.1 Introduction	67
11.2 How to Create a Story?	68
11.3 How to Use a Story?	69
11.4 Criteria	69
11.5 Example Story	71
11.6 Acknowledgements	72

Chapter 1

Introduction

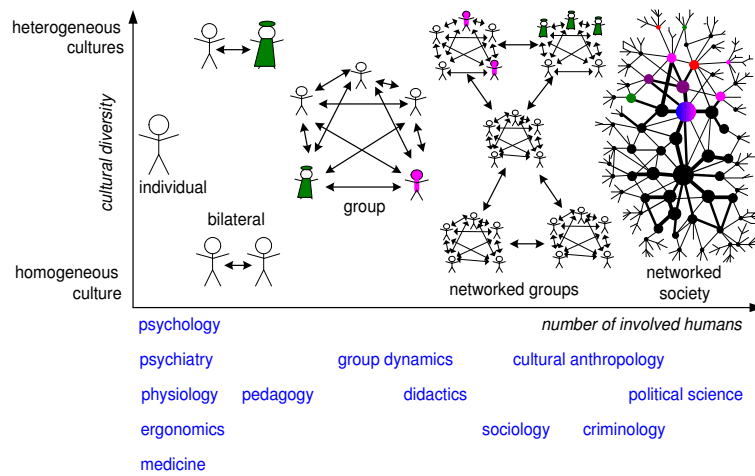


Figure 1.1: Overview of Human Aspects

Chapter 2

Human Factors in Defense

2.1 Human Factors in Defense

The defense industry is one of the major drivers behind systems engineering and architecture frameworks, see for instance [13]. These frameworks tend to focus on more technical and operational aspects, while human factors do not yet get much attention. However, the defense industry and their sponsors are changing.



Figure 2.1: Human Systems Integration DoD Acquisition

For example, the acquisition group of the Department of Defense (DoD) published this presentation [7] and this instruction [6]. Figure 2.1 is taken from this presentation and shows an inventarization of human factors to be taken into account.

1. Mission Analysis
2. Requirements Analysis
3. Function Analysis
4. Function Allocation
5. Task Design and Analysis
6. Human Interface and Team Development
7. Performance, Workload, and Training Level Estimation
8. User and Requirements Reviews

from ONR (Office of Naval Research)/SC-21 Manning Affordability Initiative
[www.hf.faa.gov/docs/508/docs/Human_System_Engineering_\(NSWC\).pdf](http://www.hf.faa.gov/docs/508/docs/Human_System_Engineering_(NSWC).pdf)

Figure 2.2: Human Engineering from Naval Perspective

The different divisions of defense are working on the human factors, see for instance the navy document [16]. Figure 2.2 shows the Navy perspective.

- HV-A: Personnel Availability
- HV-B: Quality Objectives and Metrics
- HV-C: Human Interaction Structure
- HV-D: Organisation
- HV-E: Human Functions and Tasks
- HV-F: Roles and Competencies
- HV-G: Dynamic Drivers of Human Behaviour

from The Human View Handbook for MODAF
www.hfidtc.com/ModAF/HV_Handbook_First_Issue.pdf

Figure 2.3: Human Views for MODAF

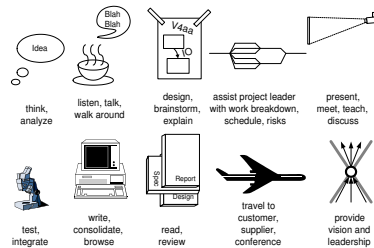
British defense is also active, as can be read in [18]. In this document a set of complimenting views is proposed for MODAF, see Figure 2.3.

2.2 Acknowledgements

Rolf Siegers (Raytheon) provided pointers to relevant information.

Chapter 3

The Role and Task of the System Architect



3.1 Introduction

Architects and organizations are often struggling with the role of the system architect (or software architect or any other kind of architect). This struggle is partially caused by the intangible nature of the responsibilities of the architect. At the other hand (good) architects are highly appreciated, even if their quantifiable output is low.

This article starts with specific deliverables, then discusses the more abstract responsibilities and, finally, discusses the day to day activities of an architect.

The role of the software architect is nicely discussed in [3].

3.2 Deliverables of the System Architect

We start at looking for the tangible output that is expected from architects. Project leaders and program managers do expect deliverables to be finished at appropriate milestones. Most Product Creation Processes define the deliverables of a System Architect to be artifacts such as documents or models. These artifacts are symbolized by the stack in Figure 3.1.

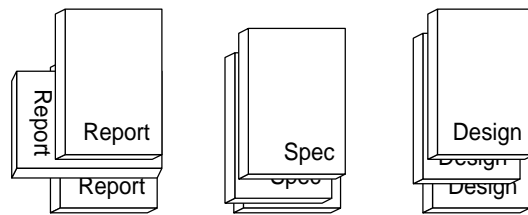


Figure 3.1: Deliverables of a system architect consists of artifacts forming a stack of paper when printed

Figure 3.2 shows the main deliverables of a System Architect more specific. Quite often the System Architect does not even produce all deliverables mentioned here, but the architect does take the responsibility for these deliverables by coordinating and integrating contributions of others. Note that some of these deliverables are part of the Policy and Planning Process.

Customer and Life-Cycle Needs	<i>(what is needed)</i>
System Specification	<i>(what will be realized)</i>
Design Specification	<i>(how the system will be realized)</i>
Verification Specification	<i>(how the system will be verified)</i>
Verification Report	<i>(the result of the verification)</i>
Feasibility Report	<i>(the results of a feasibility study)</i>
Roadmap	

Figure 3.2: More specific list of deliverables of a System Architect

3.3 System Architect Responsibilities

The System Architect has a limited set of primary responsibilities, as visualized in figure 3.3. The primary responsibilities are:

Balance of system properties as well as internal design properties. The system should be balanced: for example, the cost of subsystems should correspond with its added value in terms of functionality and performance. Architecting is a continuous balancing act in many incomparable dimensions and quantities.

Consistency across many organizational and design boundaries; From needs to implementation details, from system level to detailed implementation.

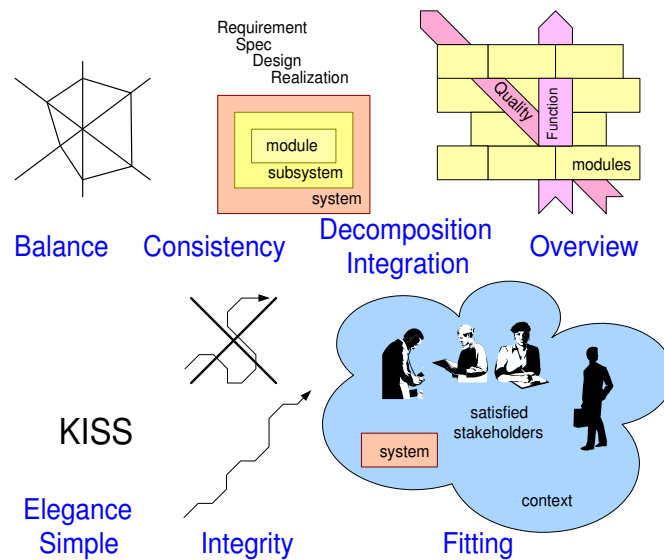


Figure 3.3: The primary responsibilities of the system architect are not tangible and easily measurable

Decomposition, Integration Decomposition is the standard answer in dealing with complex and big problems. Decomposing Systems in subsystems, subsystems in modules et cetera is a major responsibility of the architect. In most systems many decomposition dimensions are required: physical, logical, functional, and many more, see [12]. The complementary action of decomposition, however, is integration. The integral functioning and performance of the system is the ultimate goal of product creation, which emphasizes the importance of integration. In practice integration is much more difficult than decomposition, in fact the architect must decompose in such a way that integration is feasible.

Overview of the entire system and its context helps to make sensible specification and design decisions. The architect should provide overview to all members of the product creation team. Most of these members have a very limited horizon. The architect should help them by providing proper context information to make local design decisions.

Elegance, Simplicity are properties of a “good” architecture. The dangerous aspect of this responsibility is the highly subjective nature of elegance and simplicity. The appreciation of simplicity and elegance should be assessed or acknowledged by others than the architect.

Integrity of the system specification and design over time. The focus of a devel-

opment team is often wandering over time, sometimes it depends on the hype of the week. The architect is responsible for maintaining a balanced and focused development over time. For instance, when cost price reduction is required then the architect should keep performance and reliability on the agenda.

Fitting in stakeholder needs and system context, during the entire life cycle, is one of the core responsibilities of the architect. The architect must connect depth knowledge with breadth knowledge.

We can condense the primary responsibility of the System Architect as: to ensure the good functioning of the System Architecting Process. In practice, this responsibility is often shared by a team of System Architects, with one chief architect taking the overall responsibility.

responsibility	primary owner
business plan, profit	business manager
schedule, resources	project leader
market, salability	marketing manager
technology	technology manager
process, people	line manager
detailed designs	engineers

Figure 3.4: (Incomplete) list of secondary responsibilities of the system architect and the related primary owner

The list of primary responsibilities as discussed above is suffering from a lack of measurability and is rather intangible. Systems Architects also have secondary responsibilities, where these are primarily owned by other persons. Most other roles in product creation are much sharper defined, as shown in Figure 3.4. For instance the business manager is responsible for the business plan and the financial results. The project leader is responsible for the schedule and hence for completing the project in time and within budget. The marketing manager is responsible for addressing the relevant markets and hence for market share and salability of the product. The technology manager is responsible for the timely availability of technologies and related tools. The line manager is responsible for the availability of the right people, with skills and processes to do their job. Final example are the engineers who are responsible for the design of their component or module.

3.4 What does the System Architect do?

Figure 3.5 shows the variety of activities of the day to day work of a system architect. A large amount of time is spent in gathering, filtering, processing and discussing detailed data in an informal setting. These activities are complemented by more formal activities like meetings, visits, reviews et cetera.

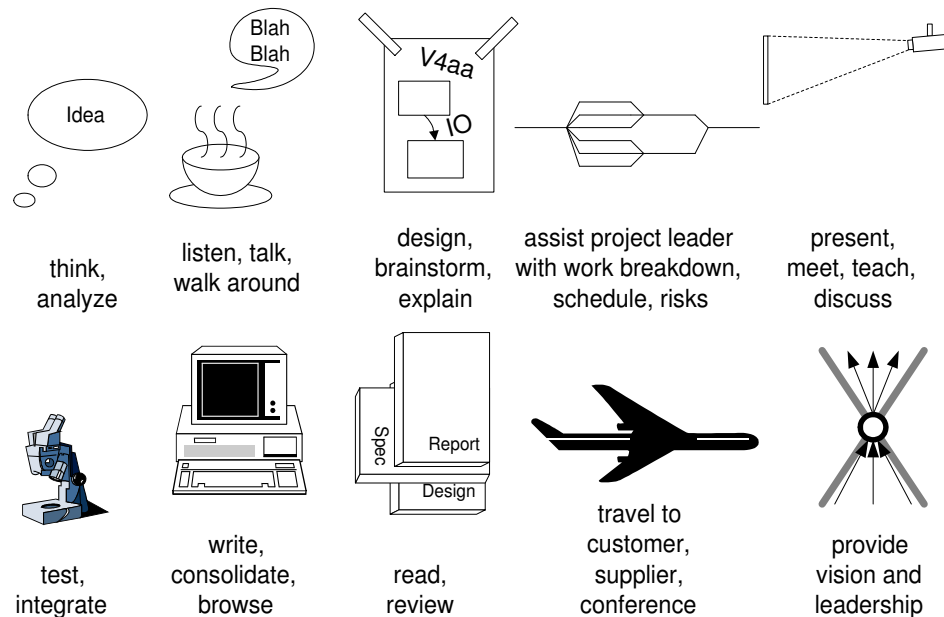


Figure 3.5: The System Architect performs a large amount of activities, where most of the activities are barely visible for the environment, while they are crucial for the functioning of architects

The system architect is rapidly switching between specific detailed views and abstract higher level views. The concurrent development of these views is a key characteristic of the way a system architect works.

Abstractions only exist for concrete facts

System Architects which stay too long at "high" abstraction levels drift away from reality, by creating their own virtual reality.

Figure 3.6 shows the bottom up elicitation of higher level views. A system architect sees a tremendous amount of details, most of these details are skipped, a smaller amount is analyzed or discussed. A small subset of these discussed details is shared as an issue with a broader team of designers and architects. Finally, the system architect consolidates the outcome in a limited set of views. The order of magnitude numbers cover the activities in one year.

The opposite flow in 3.6 is the implementation of many of the responsibilities

	Quantity per year (order-of- magnitude)	architect time per item	
consolidation in deliverables	driving views	10	100 h
	shared issues	10^2	1 h
meetings	touched details	10^4	0.5 – 10 min
informal contacts	seen details	$10^5 - 10^6$	0.1 – 1 sec
sampling scanning	product details	$10^7 - 10^{10}$	
	real-world facts	infinite	

Figure 3.6: Bottom up elicitation of high level views

of the system architect. By providing overview, insight and fact-based direction a simple, elegant, balanced and consistent design will crystalize, where the integrity of designs goals and solutions are maintained during the project.

A lot of time spent by the architect serves the purpose of communication between many project members. The architect not only responsible for the system integration, but has also an integrating role in the project itself. The architect has to interact a lot with all the people mentioned in Figure 3.4, in order to fulfil the architect's responsibilities.

3.5 Task versus Role

The task of the system architect is to generate the agreed deliverables, see section 3.2 This measurable output is requested and tracked by the related managers: project leaders and the line managers. Many managers appreciate their architects only for this visible subset of their work.

The deliverables are only one of the means to fulfil the System Architect Responsibilities, as described in section 3.3. The system architect is doing a lot of nearly invisible work to achieve the system level goals, his primary responsibility. This work is described in section 3.4. Figure 3.7 shows this as a pyramid or iceberg: the top is clearly visible, the majority of the work is hidden in the bottom.

3.6 Acknowledgements

Nicolette Yovanof pointed out that the text belonging to Figure 2 and Table 2 was rather incomplete. She also mentioned that some more attention for the interaction with non-architects would be helpful. Chuck Kilmer provided feedback

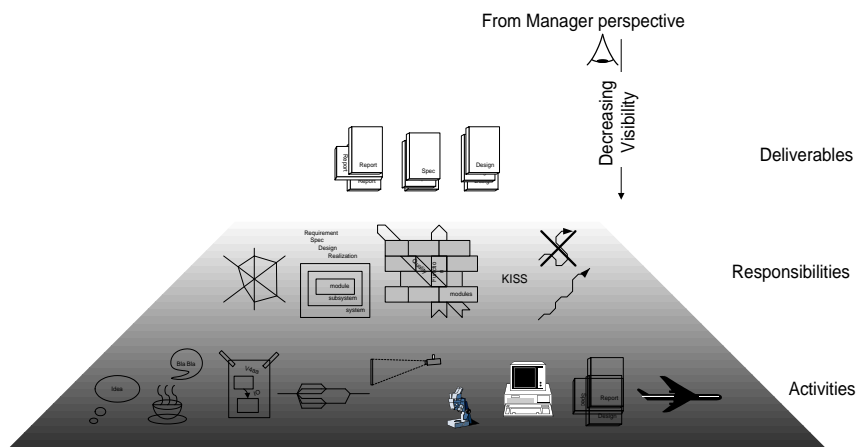
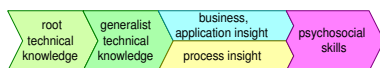


Figure 3.7: The visible outputs versus the (nearly) invisible work at the bottom

on "The Awakening of a System Architect", which resulted also in an update of this paper. Byeong Ho Gong suggested a better coverage of the interfacing with customers/stakeholders. Pierre van de Laar provided textual improvements.

Chapter 4

The Awakening of a System Architect



4.1 Introduction

System architects are very rare commodity. This chapter describes the observed general growth pattern of system architects. We hope that by analysis of the the characteristics of existing system architects will facilitate the training of new system architects. Reference [17] contains a good description of a system architect.

4.2 The Development of a System Architect

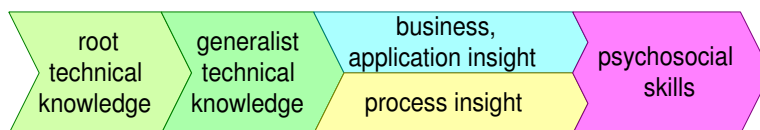


Figure 4.1: Typical Development of a System Architect

System architects need a wide range of knowledge, skills and experience to be effective. Figure 4.1 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. Section 4.3 describes the path from a mono-

disciplinary specialist to a multi-disciplinary system architect with broad technological knowledge.

When the awakening system architect has reached technological breadth, then it will become obvious that most encountered problems have a root cause outside of technology. The system architect starts to develop along two main parallel streams:

The business side: the market, customers, value, competition, logistics, service aspects

The process side: who is doing what and why, necessitated by the amount of involved stakeholders

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.

4.3 Generalist versus Specialist

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.

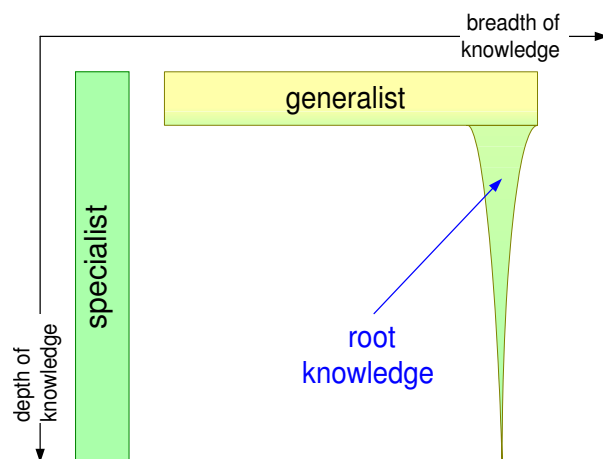


Figure 4.2: Generalist versus Specialist; depth versus breadth

Figure 4.2 is a visualization of the difference between a specialist and a generalist. 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 roots in detailed technical knowledge. These roots are important for the generalist self, since it provides an anchor and a frame of reference. It is also vital in the communication with other specialists, because it gives the generalist credibility.

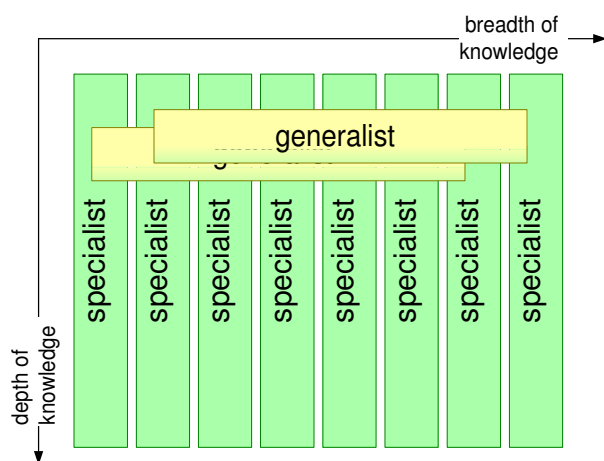


Figure 4.3: Generalists and Specialists are both needed in complex products, they have complementary expertise

Figure 4.3 shows that 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 that benefit from a generalist profile. For instance the functions of project-leader or tester both require a broad area of know how.

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 4.4 shows a more gradual spectrum from specialist to system architect. The arrows show that intermediate functions exist in larger product developments, forming natural stepping stones for the awakening architect.

Examples of aspect architects are:

subsystem architects subsystems are the main organizational decomposition. In hardware intensive systems subsystems tend to be physical, e.g. loader or generator. Typical number of subsystems is between 5 and 15.

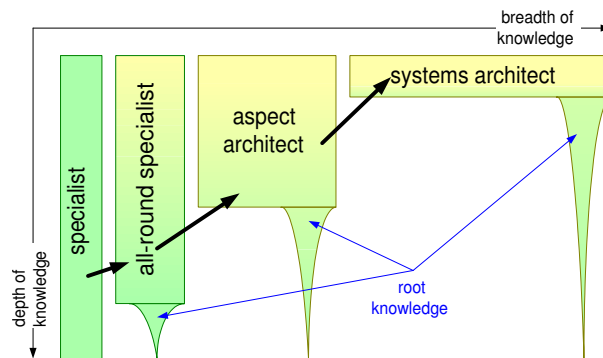


Figure 4.4: Growth in technical breadth, intermediate functions from specialist to system architect

SW, mechanics or electronics architects or discipline oriented architects. The architects ensure consistency across physical subsystems

function architects take responsibility for one system function, ensuring the soundness of that function.

quality architects take responsibility for one quality, e.g. safety, reliability, security.

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. The limited scope of one subsystem reduces the required breadth for the subsystem architect to a hopefully realistic level.

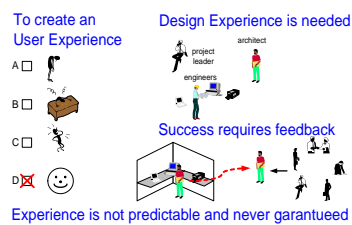
Many products are becoming so complex that a single architect is not capable of covering the entire breadth of the required detailed knowledge areas. In those cases a team of architects is required, where the architects are complementing each other in knowledge and skills. It is recommended that those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

4.4 Acknowledgements

Chuck Kilmer suggested a new title and offered many textual improvements.

Chapter 5

Architecting for Humans; How to Transfer Experience?



5.1 Introduction

Many modern appliances cause an alienated feeling for (less-technical) consumers. Figure 5.1 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.

- A [Icon of a person slumped over] depressed
- B [Icon of a person at a desk] desparate
- C [Icon of a person with a lightning bolt] hysteric

Figure 5.1: 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 5.2.

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.

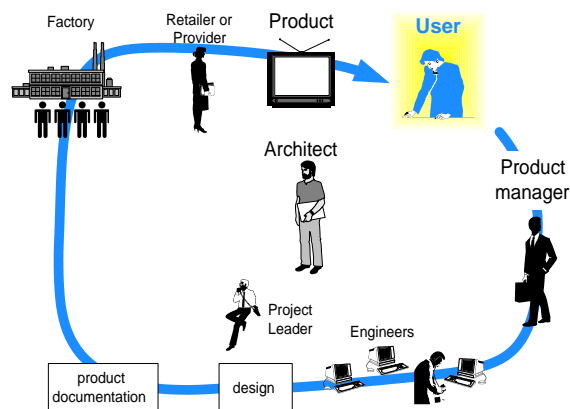


Figure 5.2: Product Creation Cycle

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 5.3 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 [11] for a further discussion on the relation between "fuzzy" user needs and SMART engineering. Figure 5.4 visualizes the gap between the user experience and the engineering world, which is to bridged by an architecting effort.

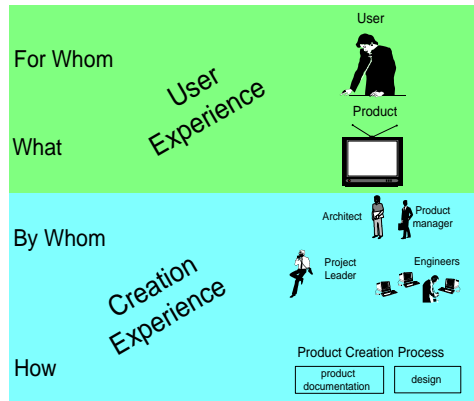


Figure 5.3: 2 Levels of Experience

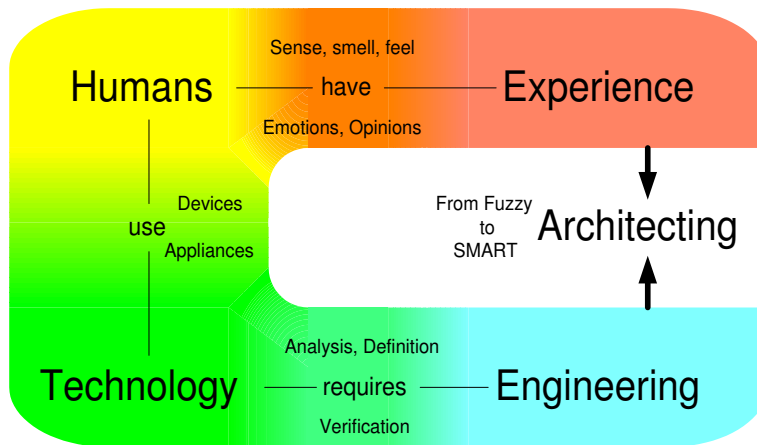


Figure 5.4: Bridging the gap between Experience and Engineering

5.2 User Experience

As an example of user experience Time shift recording is used. Figure 5.5 shows the concurrent activities that occur when straightforward time shifting is used. In this example the user is watching a movie, which is broadcast 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 of, while in the background the recording of the not yet finished movie continues.

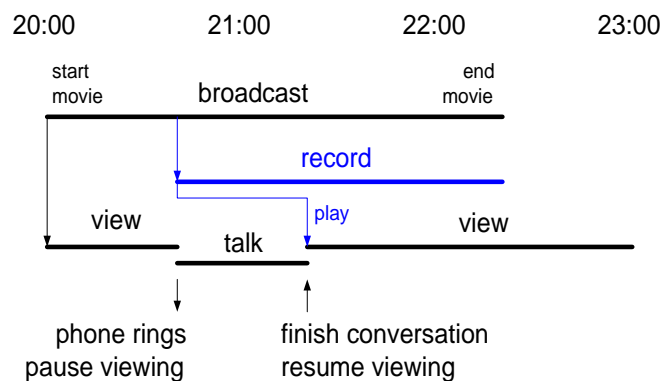


Figure 5.5: Example Time Shift recording

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 5.1 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 5.1: *Construction limits intrude in Experience*

Construction limits, but also more extensive user stories, see figure 5.6, 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.

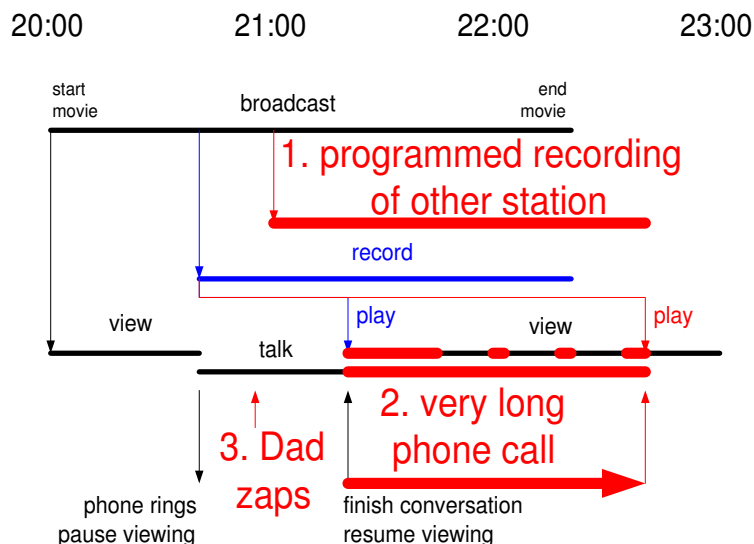


Figure 5.6: What if?

The story behind figure 5.6 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.

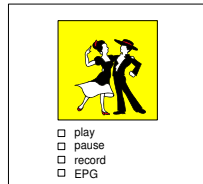
The big questions in this story is: *What is recorded when?* and *Who is able to watch the desired content later this evening?*. Most Personal Video Recorders have only one tuner and will therefore not support the three persons satisfactory.

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 5.7.

The user experience is influenced by many factors, ranging from environmental factors, such as social status, location and time to personal factors, such

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

5.1 Software Requirements	
5.1.1 Real-time data requirements	5.1.1.1 Access to the non-real-time data must be done in such a way that it does not interfere with the real-time data. 5.1.1.2 There must be no disruptions in output of video signal during the operation of VCR. 5.1.1.3 Responsiveness for open real-time data is less than 100ms from the writing a block on PECO for 200 of non-video data.
5.1.2 Implementation detail	5.1.2.1 Management of PECO content must only be possible through the TCC in order to prevent unauthorized access to content of PECO. 5.1.2.2 Visual feedback is provided to the user via On-Screen Display. 5.1.2.3 User input is provided via the RC.
5.1.3 Non-real time data requirements	5.1.3.1 User must be able to pause and un-pause a title played from PECO while in the watching or... 5.1.3.2 User can jump forward and backward in a title from PECO during watching of this title. 5.1.3.3 Names of titles should be derived from the information from the EPG pages of the program to be recorded, time and date of registration.

Figure 5.7: OOTI workshop 2001

as education, preferences and physical status. This wide variation of influencing factors is shown in figure 5.8.

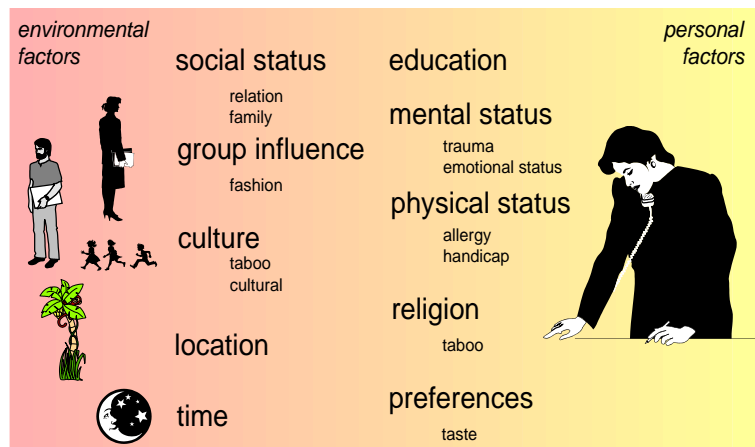


Figure 5.8: Factors influencing the User Experience

The challenge is to make the user experience more tangible, for instance by "SMART"ening the experience. Table 5.2 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 5.9 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 impossi-

- define
- measure
- predict
- verify

Table 5.2: *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		∞

Figure 5.9: Infinite Experience Space

bility to design good products, it is not that bad:

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 5.10 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.

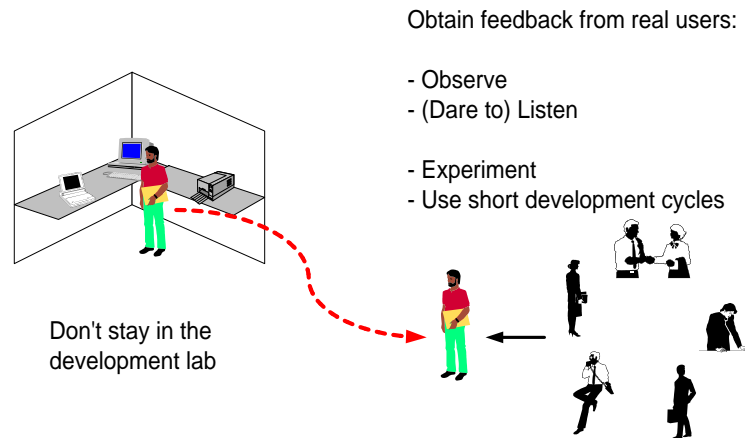


Figure 5.10: Key Success Factor: Feedback

5.3 Engineering

The world of engineering and construction is full of technologies and tools. Figure 5.11 shows some commonly used elements of this world.

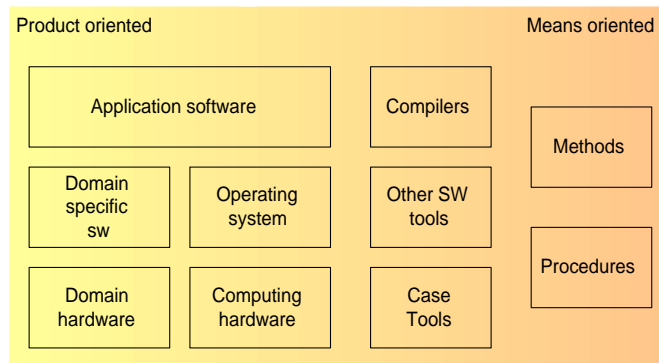


Figure 5.11: The world of the construction

The engineers are educated in construction disciplines: how to apply technologies to realize a solution which fits the specified requirements. Table 5.3 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 5.3: *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 5.12 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.

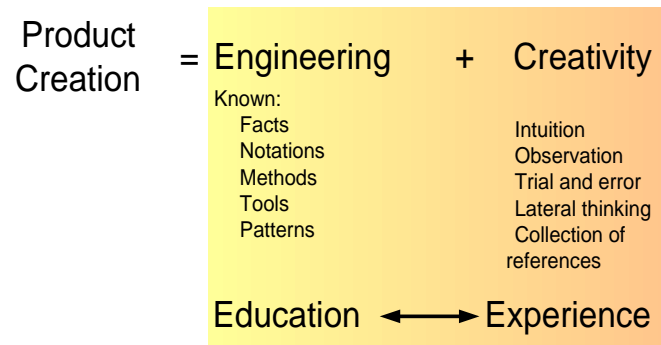


Figure 5.12: Product Creation is much more than Engineering

5.4 Education

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

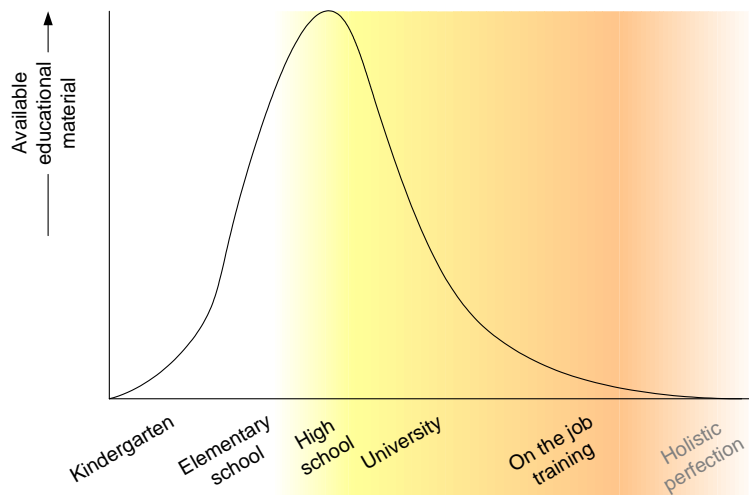


Figure 5.13: Educational Material per education stage

Figure 5.13 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	apprenticeship	Peer coaching
Interact and Listen	Lectures: Explain Show examples		Seminars Workshops Conferences	
Read	Handbook Course material		Magazines Journals	

time →

Figure 5.14: Changing Education model in time

The education can be decomposed in 3 types of learning: *doing*, *reading* and *interacting and listening*, see figure 5.14. 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 5.15: Increasing Initiative required

Figure 5.15 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 5.4: *Prerequisites for continuous successful product creation*

Table 5.4 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.

5.5 Conclusion

Figures 5.16 and 5.17 summarize the article.

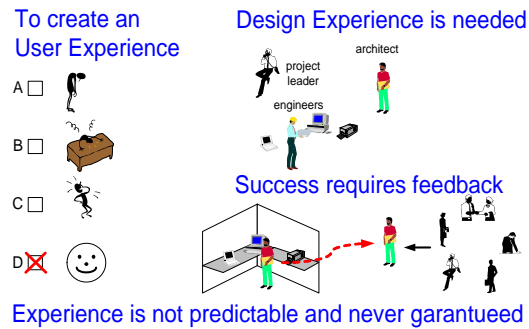


Figure 5.16: Architecting for Humans

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.

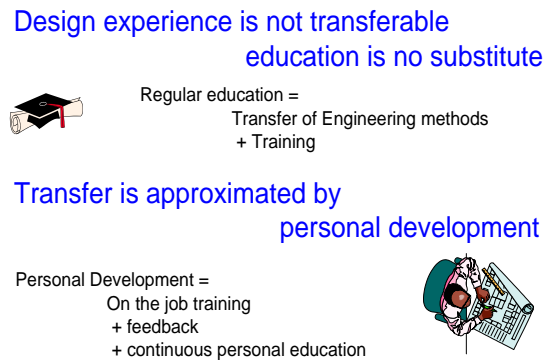


Figure 5.17: Experience Transfer

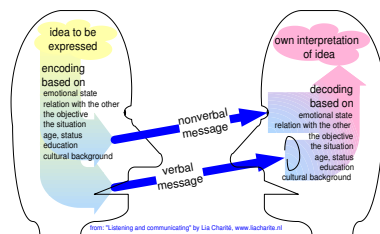
Design experience itself is not transferable, education is a means which can enable in potential good engineers to build up experience faster. This requires a lot of practical training. Later on this engineer must continue his personal development, by means of on the job training, feedback and peer coaching.

5.6 Acknowledgements

Daniel Malacarne provided feedback on a number of figures.

Chapter 6

Human Side: Interpersonal Skills



6.1 Introduction

We often take for granted that two individuals can cooperate. However, in practice many problems arise at the fundamental level of cooperation between two individuals. We will first discuss the wonder of communication, since communication is the starting point for cooperation. Next we will discuss a set of techniques that can be deployed between two (and often more) individuals:

- investigation and acknowledgement
- constructive feedback
- conflict management
- appraisal
- good practices in a conversation
- searching for ideas

6.2 The Wonder of Communication

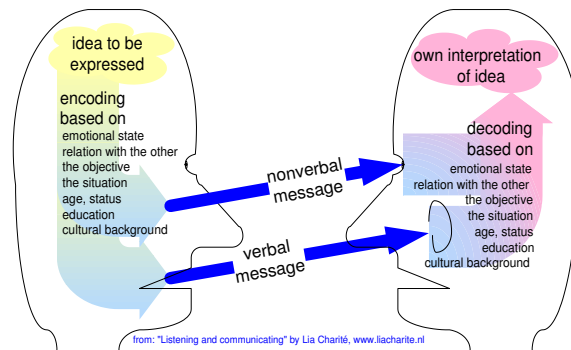


Figure 6.1: Active listening: the art of the receiver to decode the message

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 6.1. The receiver of this message has to decode this message and interprets the message, however, based on many similar 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, humans, are able to convey messages to others.

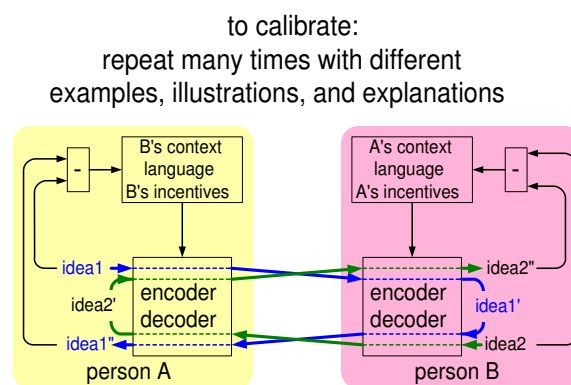


Figure 6.2: Intense interaction needed for mutual understanding

The mechanism behind this miracle can be understood by extending the model of sender and receiver as in figure 6.2. 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.

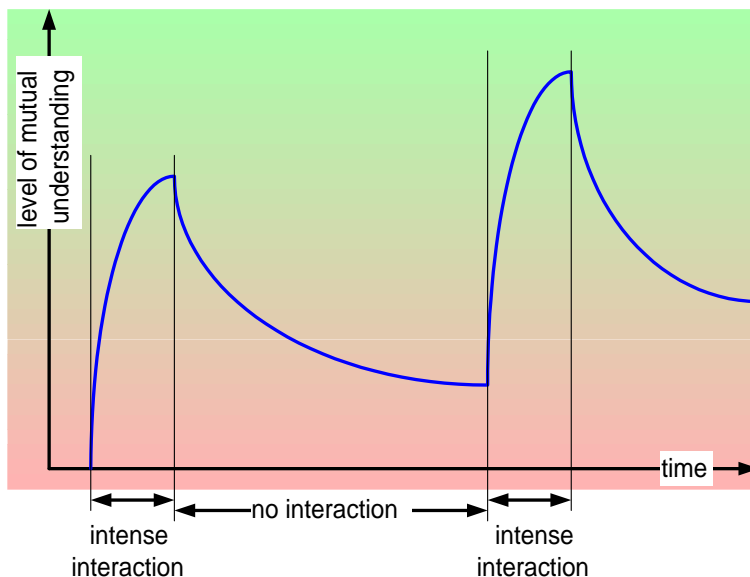


Figure 6.3: Mutual understanding as function of time

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, since continuous calibration takes place. Without interaction the mutual understanding degrades, due to the dynamics of the interpretation. Figure 6.3 visualizes the mutual understanding as function of time and interaction.

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 a minor factor¹ in comparison with the human differences which have to be bridged continuously.

¹Dogmatic applied unification of terms and notations works often counterproductive. Problems or viewpoints might be more easily expressed in other terms, while the unification drive blocks the search for a mutually understandable expression. Active participation is required to obtain understanding.

6.3 Interpersonal skills

In the previous section we explained that it is rather miraculous that communication between individuals works. In this section we provide a number of fundamental skills to facilitate the cooperation between two individuals.

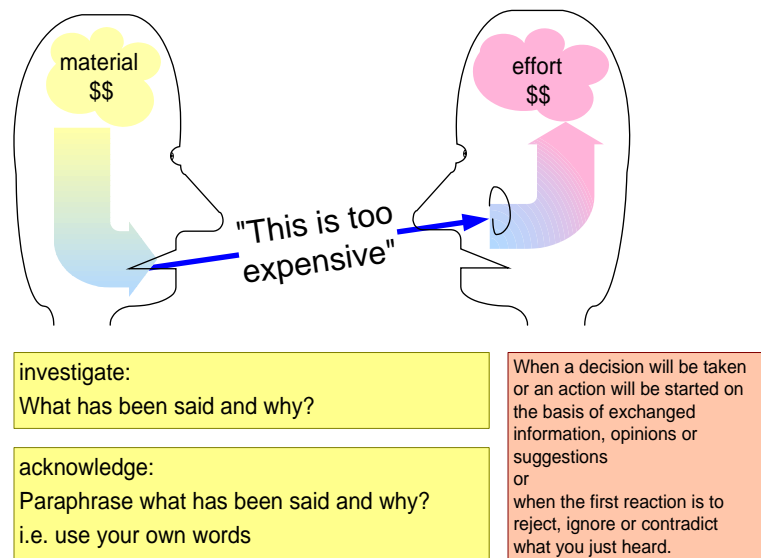


Figure 6.4: Investigate and Acknowledge

Investigate and Acknowledge, see Figure 6.4 is a technique that helps to bridge the differences between individuals. Investigation is the process of asking questions to ensure that you understand what the other person is saying. Acknowledgement is the process to help the other person understand what you have in mind.

In practice people tend to make decisions after a superficial understanding of what someone else has said. The risk is that both persons did not understand each other, with the risk that the decision is faulty. Also tensions between persons can be traced back to a lack of mutual understanding.

Note that application of the investigate and acknowledge technique in first instance slows down the communication. Its application should not be overdone, this may also cause irritation.

The basis for improvement and learning is **feedback**, get observations about the performance and its consequences from someone else. Key for effective feedback is that it is constructive. This means that not only the poor performance issues should be mentioned, but also the strong points, and that the strong points are related to ways to improve the weak points. This is called *constructive feedback*, see Figure 6.5.

Conflicts are part of normal human interaction. Conflicts in itself are not bad.

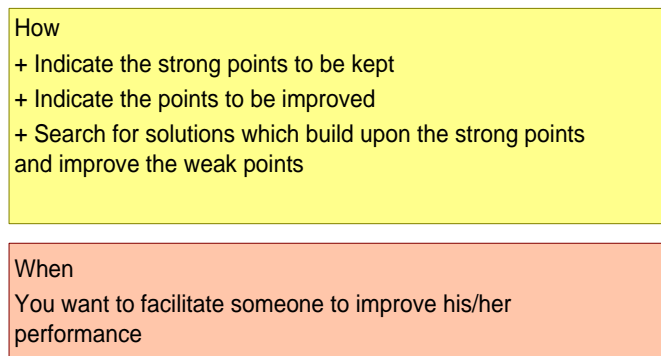


Figure 6.5: Constructive Feedback

However, conflicts that are not handled can grow to unwanted dimensions. **Conflict management** is a technique to explicitly cope with conflicts, see Figure 6.6

The first step in conflict management is to understand the conflict itself. The conflict can be made explicit by formulating what is important for you and why, and by investigation and acknowledgement of what is important for the other and why. A fundamental choice has to be made when the positions are clear:

- If you are willing and able to consider alternatives, then you can jointly search for alternative solutions.
- If you are not willing and able to consider alternatives, or no acceptable solution for both parties can be found, then the conversation must be finished by acknowledging the right to have a different opinion, and by indicating your decision and its rationale.

The main message is that "false" hope should be avoided, rather the conflict should be clearly finished one way or the other. The risk of maintaining false hope is that the conflict will simmer on and it might become a festering disease.

Appraisal can work wonders for the motivation of people. Nevertheless, many people don't know or are hesitant to give appraisal. Figure 6.7 shows when and how to appraise. Essential for appraisal to work is that the appraisal is authentic. In the eyes of the person giving the appraisal the performance of the the appraised person should be so good that appraisal is justified.

The appraisal itself must be specific. Some people appraise simply by stating that "you did a good job". Such generic appraisal does not really help the appraised person. Rather the appraisal should explain:

- what the person did specifically
- what personal qualities were applied that enabled this performance
- and what benefits this work had for other people or the organization

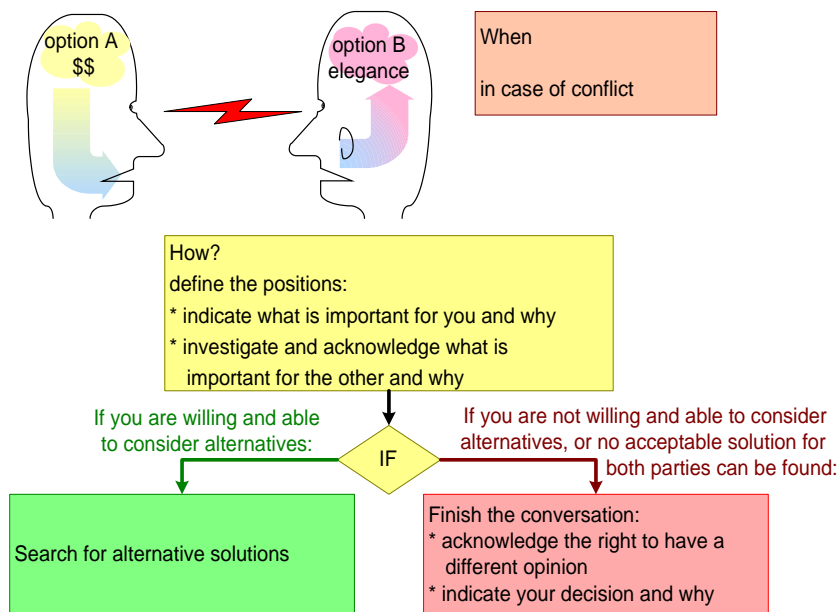


Figure 6.6: Conflict Management

Very elementary for effective communication is that **conversations** are opened and closed clearly. Figure 6.8 shows a number of good practices for opening and closing a conversation. At the beginning of a conversation it is good to make explicit what the purpose of the conversation is. At the end it is good to summarize conclusions and agreements, for example in terms of an action plan.

In several circumstances it can be desired to **search for new ideas** or alternatives. Figure 6.9 shows several recommendations for searching ideas. The basic idea is that two persons should build upon each others ideas, when searching for ideas. They should stimulate each other to make suggestions, and to keep going the other person should give a reaction. Credit should be given to the person who originates ideas. The last recommendation to find ideas is a technique from the creativity field: people come up with new solutions if you remove or add constraints. E.g., if cost is not a problem, how would you then solve the problem? Interestingly many solutions found this way can also be used when the constraint is reapplied.

When
 Someone's performance is important for you
 * exceeding the expectations
 * meets expectations continuously
 * meets expectations, which exceed the normal performance level of this person

Appraise only when authentic!

How
 + Mention the performance very specific.
 + Mention the personal qualities which lead to this performance.
 + Describe which advantages arise for you, the department or the organization.

Figure 6.7: Appraisal

When you open a conversation
 formulate the purpose

When you finish the conversation
 summarize the agreements and the actionplan

Figure 6.8: Conversation Good Practices

When asking for a suggestion	→	give a reaction
When supplying a suggestion	→	ask for a reaction
When you use or build upon ideas of others	→	mention the source of the ideas
When you need new or more creative ideas	→	remove limitations temporarily or add limitations

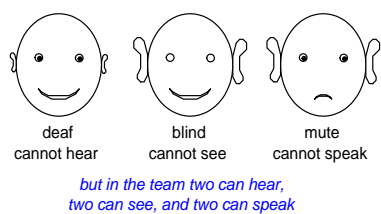
Figure 6.9: Searching for Ideas

6.4 Acknowledgements

The communication section is based on material from Lia Charite (www.liacharite.nl). The techniques for interpersonal skills are based on the course “Interpersonal Management Skills” by Hay Management Consultants.

Chapter 7

Human Side: Team Work



7.1 Why Work in Teams?

Today's product creation projects involve so many fields of expertise that we need many different specialists. Teams are a way to organize the project. From management perspective manageability is one of the main reasons to use teams.

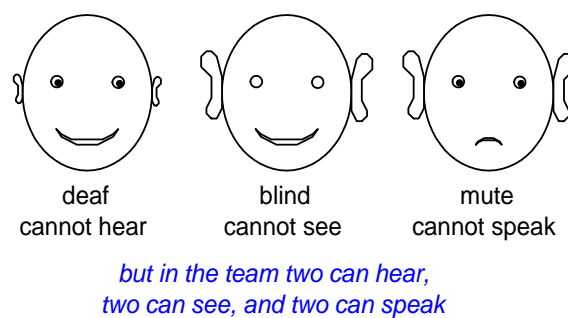


Figure 7.1: Teams consist of peoples with complementary skills and knowledge

Teams can also be an effective way to benefit more from the strengths of individuals within in a group. A well designed team is more than the constituting members. Figure 7.1 shows a schematic version of the figure of the three apes, where one ape cannot see, one ape cannot hear, and one ape cannot talk. If all three

team-up, then we have a team where two apes can see, two can hear, and two can talk. The team members are complementary and jointly they can do much more than every individual alone.

7.2 Team size

Teams can vary from very small (two people) to very large (thousands of people). However, large teams are often further divided in smaller teams, since human interaction gets more difficult with more team members. For example, an entire project is decomposed in subsystems and further decomposed in components, where the team organization follows the same decomposition.

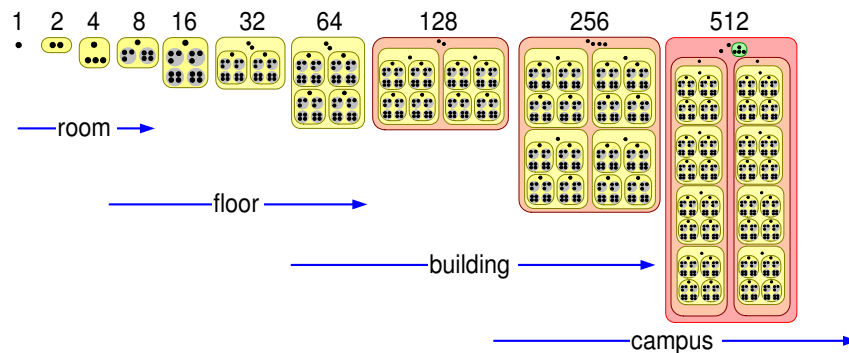


Figure 7.2: The way of working of a team depends on the team size. Every factor 2 in size creates a different paradigm.

In practice only small team, e.g. between 3 and 8 people, show intense interaction. When teams are bigger, the team de facto divides itself further to “human” scale teams. Figure 7.2 shows that large organizations are broken down in smaller units. It also shows that the size impacts the housing and location of teams.

The operation of a team depends strongly on its size. Very small teams of two or three people will interact more intense and informal than larger teams. Larger teams will need more time to communicate, lowering their efficiency. In fact every increase of the organization size with circa 50% triggers changes in the mode of operation.

Figure 7.3 shows a very simplistic model of the communication and the productivity of a team member. Every team member is modeled as being able to spend time on 4 tasks each 25% of the time. Every task is either producing something or communication with someone. This simple model shows that working with more people gets quickly less efficient due to communication overhead.

This simple model can be transformed in a hierarchical team model, see 7.4. The hierarchy reduces the communication overhead, and hence improves efficiency.

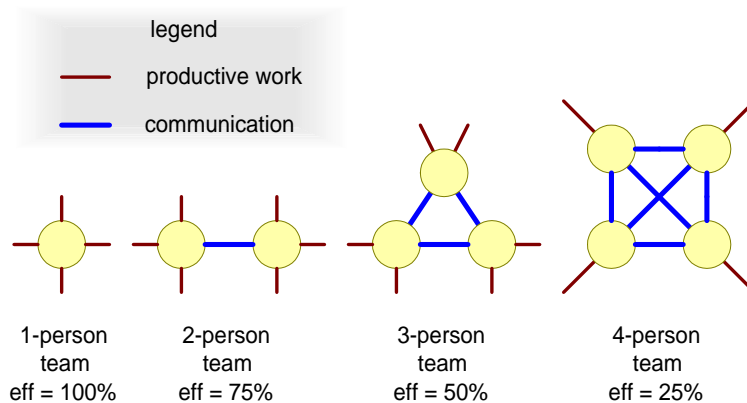


Figure 7.3: Very simplistic model of team communication and productivity

Note that the hierarchy increases length of the communication paths. Longer communication paths suffer from more latency and more deformation. Efficient organizations tend to combine a hierarchical structure with more direct communications between individuals, creating a network organization.

7.3 Team composition

A good team consists of complementary members, where the members can cooperate well. The team “chemistry” must be good. In the literature many role models are given that can be used to “design” a team.

Figure 7.5 show the roles described by Meredith Belbin [2] See for a summary of these roles [1]. People tend to have preferred role patterns that they follow. This is not black and white, people can be mostly plant and somewhat of a chairman at the same time. The idea is that a good team needs all different roles.

Another role model is provided by Edward de Bono [5], the so-called *Six Thinking Hats*. These colored hats symbolize the natural attitude that persons bring into the team. Figure 7.6 shows the six different colors and their main characteristics. Again the idea is that these different kinds of people are complementary, a team needs positive and negative oriented members, creative and process oriented members, and neutral and feeling oriented members. Also note that some members can take on all colors: chameleons.

A famous ontology is provided by the Myers-Briggs type indicators [14]. Four characteristics with their two extremes are used as parameters to classify people, following Jung’s ideas with similar characteristics. In Figure 7.7 these four characteristics are shown: *Extraversion* and *Introversion*, *Sensing* and *iNtuition*, *Thinking* and *Feeling*, and *Judging* and *Perceiving*. Someone’s personality type can be captured by concatenating the 4 letters of these characteristics. For example, it

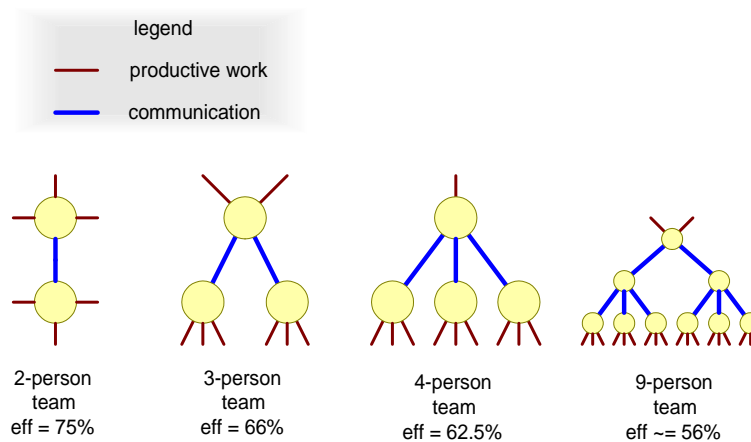


Figure 7.4: Very simplistic model of team communication and productivity with hierarchy

<i>plant</i> creative	<i>team worker</i> cooperative, averts friction	<i>implementer</i> disciplined, conservative, doer
<i>resource investigator</i> enthusiastic communicator	<i>shaper</i> driver, dynamic	<i>completer finisher</i> conscientious, painstaking
<i>coordinator</i> mature, chairman	<i>monitor evaluator</i> sober, analytical	<i>specialist</i> single-minded, rare skills

Belbin's team roles

Figure 7.5: Belbin team roles

is often observed that systems architects have INTP as personality, see for instance <http://www.e-mbti.com/intp.php>.

To design a team the roles/personalities of its intended participants have to be taken into account. However, we also have to design the team to get:

- a multitude of opinions
- coverage of the involved stakeholders
- coverage of knowledge and skills

7.4 The Process of Creating and Employing a Team

Let us assume that someone in the organization, for example the business manager or project leader, has the need for a team. We will call this person the team owner,

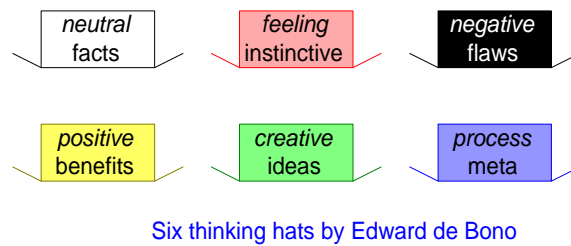


Figure 7.6: Six thinking hats by Edward de Bono

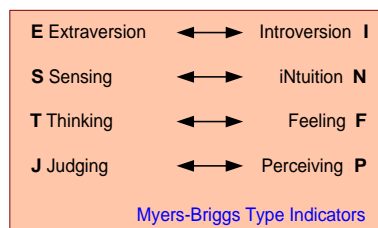


Figure 7.7: Myers-Briggs type indicators

meaning that this person feels responsible for a good functioning team and has a need for its results. Figure 7.8 shows this team owner at the top of the figure.

This team owner will compose the team as described in 7.3, will arrange facilities, such as housing as described in 7.5, and will have to provide a charter for the team. A charter provides the scoping for the team: *what* has to be done *why*, *how* to achieve this, *whom to involve*, and *when* and *where*. This charter gives direction for the team. However, the team should be able to determine its own way-of-working within the charter. Micro-management of the team by its owner or others outside the team will greatly reduce the team's productivity. In other words the team has to be empowered by the owner.

The team will produce results while working. These results have to be respected by the receivers, such as the owner. If the results are not respected, then this will discourage this team and also the successive teams: *Why engaging in team activity, if the results are not taken seriously?* The owner should consider this aspect before initiating a team. Teams might reach conclusions that are undesired by the initiators. Sometimes, teams are initiated as decoy rather than a real goal; this works once, after that employees are frustrated and it takes a long time to repair trust and motivation.

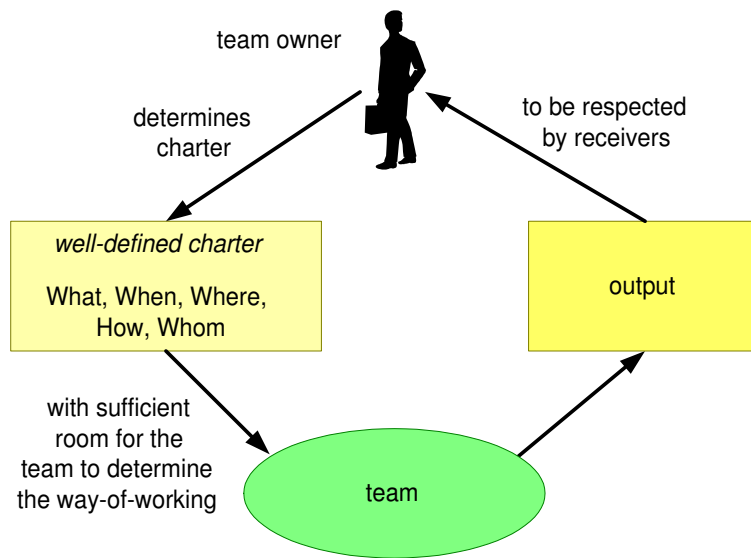


Figure 7.8: Process of creating and employing a team

7.5 Housing and Location

Housing can be used as instrument to boost team productivity and cohesion. When team members sit in one and the same room, then they will communicate more frequently and more natural.

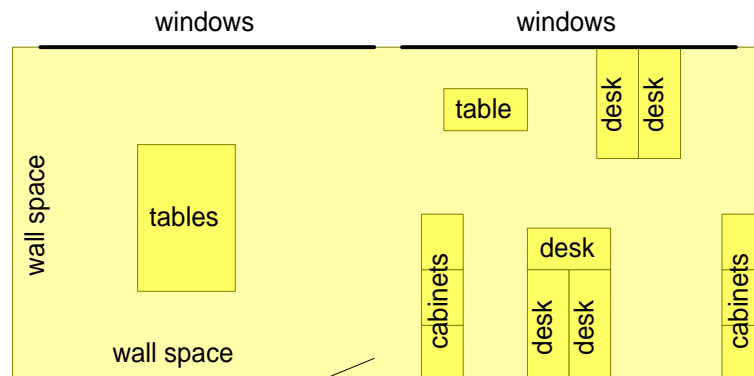


Figure 7.9: “War Room” is very effective

Figure 7.8 shows an example of a room used by the systems engineering team that designed the next generation wafer steppers at ASML. This large room had space for the normal desks with PC’s, and a meeting space with plenty of wall space. The wall space can be used for white-boards, flip-overs, and large format

print-outs of diagrams. This type of war room is very effective.

Housing is an effective means to improve team efficiency. There should not be any obstacles, such as distance, to let team members communicate. The room described above also supports communication by providing wall space and simple means to share visualizations. In LEAN product development also the physical system or the components are shared in the same location, supporting tactile and visual discussions. In a Concurrent Design Facility the room facilitates sharing of computer based models by using multiple large screens for projection.

7.6 Concurrency

Organizations seem to increase the number of activities continuously. Individual employees get more and more activities they have to do, often more or less concurrently. These activities tend to fragment the time of individuals, working a little on the first activity, do something on the second activity, continue with the next, et cetera. Figure 7.10 shows this fragmentation. It also shows that working in burst mode, e.g. working focused for one day, one week or one month on a single activity can be more efficient, because less context switches are required.

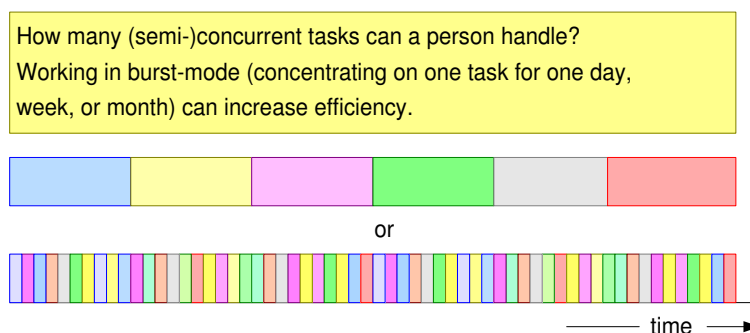


Figure 7.10: Many engineers have to divide their attention over a multitude of activities

Figure 7.11 takes this line of reasoning further. It shows six activities that are being executed in parallel and the same activities, but now executed sequentially. When the activities are done in parallel, then the results of all activities become available when all work is finished. When the activities are performed sequentially, then the first result gets available after one sixth of the time of the parallel approach. This means that the result itself, and feedback on this results becomes available much earlier. In other words, when working in parallel all results are late.

Figure 7.11 is a simplification of reality. It might be that some activities need to be parallel, for example because of the inherent elapsed time of the activities.

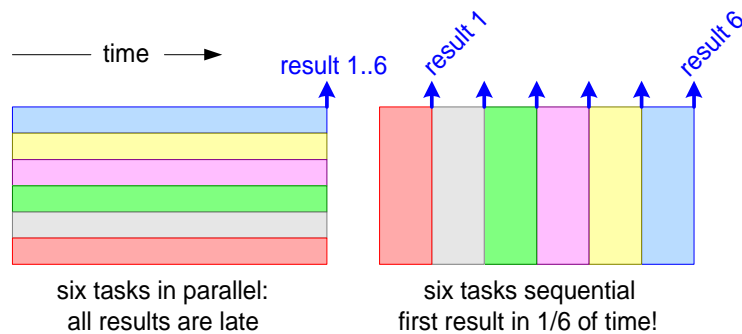


Figure 7.11: It can be more efficient to do activities sequential rather than parallel

Nine mothers cannot deliver one baby in one month, although nine mothers can deliver nine babies in nine months.

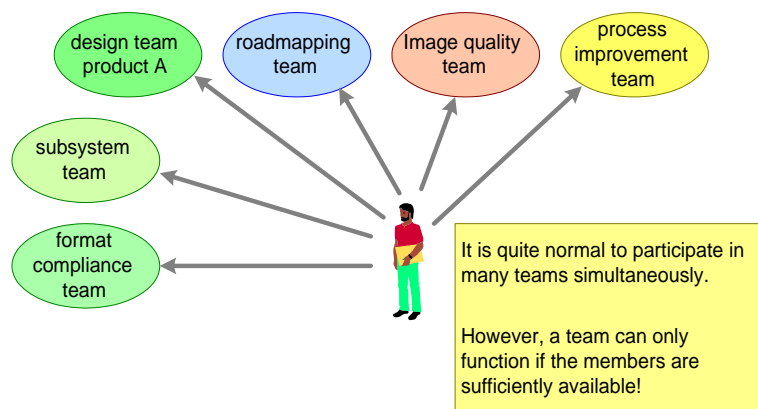


Figure 7.12: One person will be member of multiple teams

If we map these consideration on teams, then we should realize that in practice people, and certainly systems architects, participate in multiple teams at the same time. An example is shown in Figure 7.12.

The systems architect in Figure 7.12 works on the creation of a product, and is part of the *design team* of that product. At the same time the architect participates in more specific teams of that product creation, such as one of the *subsystems* and some specific aspects such as *format compliance*. However, the system architects will also participate in broader concerns of the organization, such as *image quality*, *roadmapping*, and *process improvement*.

If all these activities run concurrently, then some of them might suffer. For example, it might be better to focus for four weeks entirely on roadmapping together with the other roadmapping team members, and after that time continue with the

day-to-day architecting concerns of the product creation.

7.7 Critical success factors

Figure 7.13 shows a summary of the critical success factors that we have discussed.

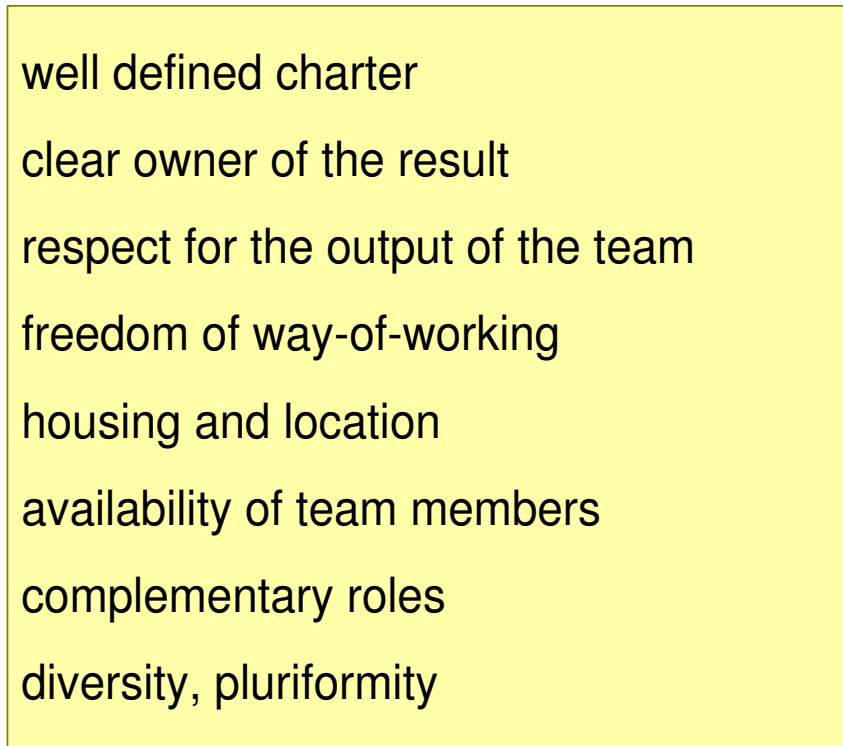


Figure 7.13: Critical Success Factors for teams

well defined charter for the team: *what, where, when, how, whom*

clear owner of the result Who needs the team, who ensures appropriate support and facilities, who is worried when the team hits obstacles?

respect for the output of the team The team with all its expertise might draw unwanted, undesired conclusions. The output of the team has to be respected, otherwise follow-up teams will not be motivated.

freedom of way-of-working The owner has to empower the team, micro-management will stifle the teams effectiveness.

housing and location are instruments to forge a team, co-location in a war room is recommended.

availability of the team members is a prerequisite to work as a team. The availability can be arranged by careful preparation and clear allocation of team members.

complementary roles so that the team is more than the sum of the individual participants.

diversity, multitude of opinions Some controversy or tension in the team is healthy to prevent inbreeding and blind spots.

7.8 Acknowledgements

Ben Spierenburg showed me very long ago the simplistic model of team communication and productivity. The sequential versus concurrent tasks diagram was shown to me by Philips Organization and Efficiency department.

Chapter 8

Architecting Interaction Styles

provocation	when in an impasse; provoke effective when used sparsely
facilitation	especially recommended when new in a field; contribute to the team, while absorbing new knowledge
leading	provide vision and direction, make choices risk: followers stop to give the needed feedback
empathic	take the viewpoint of the stakeholder acknowledge the stakeholder's feelings, needs, concerns
interviewing	investigate by asking questions
whiteboard simulation	invite a few engineers and walk through the system operation step by step
judo tactics	first listen to the stakeholder and then explain cost and alternative opportunities

8.1 Introduction

A system architect has to use different interaction styles in different circumstances. In some circumstances a *leading* style is appropriate, while in other circumstances a *facilitating* style is more effective. Figure 8.1 shows the styles that are discussed in this chapter.

8.2 Provocation

A provocative style can be used by the architect when the discussion is in an impasse. The provocation can be based on taking an extreme viewpoint of one of the stakeholders and confronting the other stakeholders with the consequences. Such a provocation forces the involved stakeholders to formulate their needs more sharp, including the consequences of following the recommendation.

A provocative style should be applied scarcely. Once team members get used to this style then the style becomes ineffective. Most people do not like to be provoked continuously, so they stop to respond after a few provocations.

provocation	when in an impasse: provoke effective when used sparsely
facilitation	especially recommended when new in a field: contribute to the team, while absorbing new knowledge
leading	provide vision and direction, make choices risk: followers stop to give the needed feedback
empathic	take the viewpoint of the stakeholder acknowledge the stakeholder's feelings, needs, concerns
interviewing	investigate by asking questions
whiteboard simulation	invite a few engineers and walk through the system operation step by step
judo tactics	first listen to the stakeholder and then explain cost and alternative opportunities

Figure 8.1: Interaction styles for architects

8.3 Facilitation

The facilitation style is a style where the architect serves the team by facilitating meetings and workshops. Facilitating a meeting means:

- preparing the meeting or workshop together with the owner of the meeting: determining the goal, participants, place, agenda, means.
- facilitating the meeting itself: timekeeping, managing the flips, writing action point and conclusions.
- finalizing the meeting: writing a report and presentation of the results, chasing follow-up actions.

The facilitation style is especially useful for architects entering a new domain. The architect provides visible value for the team, while as a spin off the architect learns a lot about the new domain.

8.4 Leading

A leading style is a style where the architect is highly visible. The architect provides vision and direction to the team. The leading architect can be recognized by looking at the followers: if they really follow the architect then the architect is effective as leader.

The risk of this style is that the team starts to trust the architect decisions too much. Most of the team members have much more know how about the design issues than the architect. The architect will often make decisions based on limited know how that should be corrected by the specialists with more know how. The

leading style sometimes inhibits the specialists to oppose the architect. The leading architect must be aware of this effect. Sometimes even invitations to oppose and provocations do not help to loosen up the followers.

8.5 Empathic

The empathic style is based on taking the viewpoint of the stakeholder under discussion. This goes much further than the objective rational view. The feelings and emotions of this stakeholder must be taken into account as well. The understanding of the state of mind is communicated back to the stakeholder. The result of this way of interacting is that the architect gets a much better insight in the stakeholder, while at the same time the stakeholder has the feeling to be taken seriously.

8.6 Interviewing

Architects pose lots of questions, questions are one of the most important instruments of the architect. The interviewing style makes excessive use of questions. The architect uses a priori knowledge to formulate open questions. These open questions must lead to an understanding of the stakeholder concerns.

The difficult part of this style is to use a priori know how in a limited and constructive way. The danger of a priori know how is that it limits observation and that suggestive questions are formulated instead of open questions.

8.7 White-board simulation

The white-board simulation style is used in meetings where a few specialists are present. The architect guides the specialists through a use case, where every specialist explains the system behavior from the specialist viewpoint. For example, the use case can be to push a *next channel* button on the user interface. In this example the user interface signal will trigger an avalanche of events in the system, going through many layers and propagating to many subsystems.

This guided simulation often reveals a lot of unknown system behavior, strange dependencies, inefficient sequences and many more engineering surprises. The normal reactions of the participants is that after a few steps they want to redesign the system. The architect should suppress this urge, by parking improvements at the side. The main purpose of this style is to build a shared understanding of the current design.

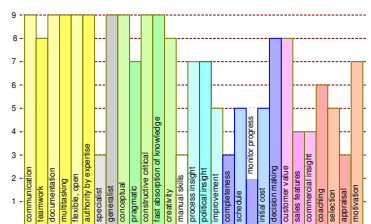
8.8 Judo tactics

The basis of judo tactics is that the architect starts to listen to the stakeholder, especially when the architect feels an urge to contradict the stakeholder. After listening to the stakeholder, and acknowledging the validity of the needs, the architect explains the costs and trade-offs. In many cases the stakeholders have a healthy feeling for value and cost and look for a reasonable balance. Quite often the result is a decision that the architect wanted to make right at the beginning. However, this style works only if the architect really listens, and is willing to take a different direction if needed. It might be that the architect discovers that the value for the stakeholder is much larger than originally assumed!

In many cases ill communication and bad listening skills block reasonable decisions. The judo style, where the architect starts to listen, avoids this trap.

Chapter 9

Function Profiles; The Sheep with Seven Legs



9.1 Introduction

Many human resource and line managers struggle with the questions:

- What people have the potential to become good system architects?
- How to select (potential) system architects?

Employees thinking about their careers might similarly wonder if they have the capabilities to become a good systems architect.

We list a number of characteristics of individual humans. We map these characteristics on different jobs, such as system architect, developer, and line manager, indicating the relative importance of this characteristic for that job. We first discuss the different jobs and their typical characteristics in 9.2 to 9.7. Then we elaborate the characteristics in 9.8.

The attention for this subject is increasing. Recent research is being carried out by Keith Frampton, see amongst others [8].

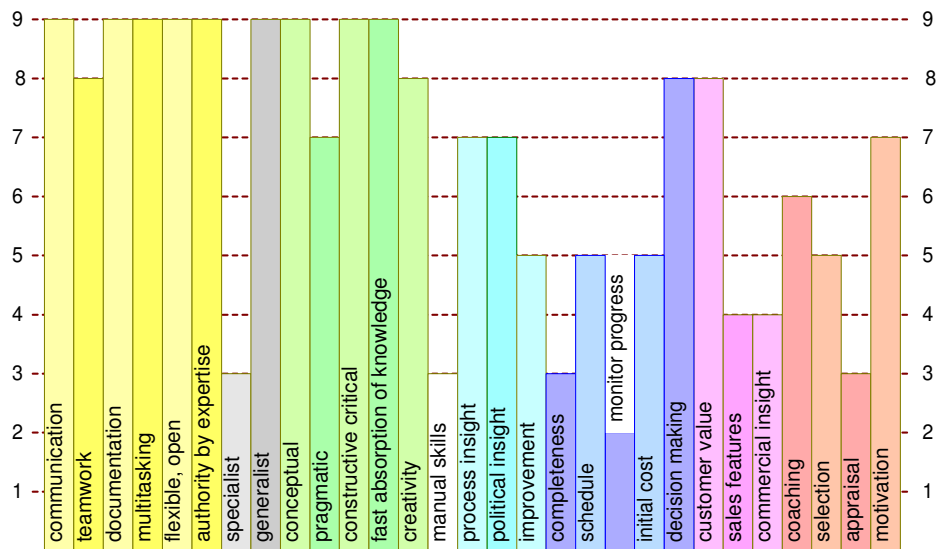


Figure 9.1: The function profile of the systems architect

9.2 Systems Architect Profile

The profile of the “ideal” system architect shows a broad spectrum of required skills. 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 of technical leadership, where the architect provides direction (*know-how* and *reasoning power*) as well as moderates the integration (*interpersonal skills*).

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 of such ideal architect allows for the formation of architecture teams. Such a team of architects can come close to this profile.

9.2.1 Most discriminating characteristics

In practice the following characteristics are quite discriminating when selecting (potential) systems architects:

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

Generalist The first reduction step is to select the *generalists only*, reducing the input stream with one order of magnitude. The majority of people feels more comfortable in the specialist role.

Multi-tasking The next step is to detect those people that need undisturbed time and concentration to make progress. These people become unnerved in the job of the systems 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 the people with poor multi-tasking capabilities become systems architect, then they are in severe danger of stress and burn out. Hence it is also the benefit to the person self to assess the multi-tasking characteristic fairly.

Authority by expertise The attitude of the (potential) architect is important for the long term effectiveness. Architects who work on the basis of delegated *power* instead of *authority by expertise* 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 (e.g. *authority by expertise*) costs a lot of energy, but it pays back in the long term.

Conceptual thinking and pragmatic 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.

9.3 Test Engineer Profile

The *test engineer* function at system level requires someone who *feels* and *understands* the system. Test engineers are capable of operating the system fluently and know its quirks inside out.

The main difference between an architect and a test engineer is the different balance between **conceptual thinking** and **practical doing**. Test engineers often have an excellent intuitive understanding of the system, however they lack the conceptual expression power and the communication skills to use this understanding pro-active, for instance to lead the design team.

9.4 Developer Profile

The core value of developers is their specific discipline know-how. Good developers excel in a limited set of specialties, knowing all tricks of the trade. On top

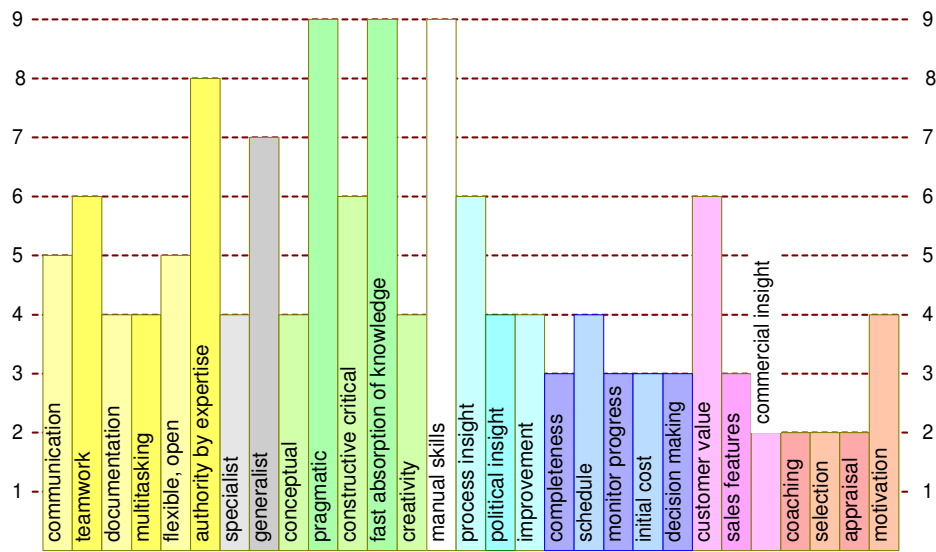


Figure 9.2: The function profile of the test engineer

of this they should be able to deploy this know-how in a creative way. In today's large development teams a reasonable amount of *interpersonal skills* are required as well as *reasoning power* and *project management* skills.

9.5 Operational Leader Profile

The *operational leader*, for instance a project leader, is totally focused on the result. This requires *project management* skills, the core discipline for operational leaders.

The *multi-tasking* capability is an important prerequisite for the operational leader too. If this capability is missing the person runs a severe risk of getting a burn out.

Note also that the operational leader functions as kind of gatekeeper, where the *completeness* is important.

9.6 Line Manager Profile

The *line manager* manages the intangible assets of an organization: the people, the technology and the processes. Technology and process know-how are tightly coupled with people, this know-how largely resides in people and is deployed by people. *Human resource management* skills and *process* skills are the core discipline for line managers, which need to be supported with sufficient *specialist* know-how.

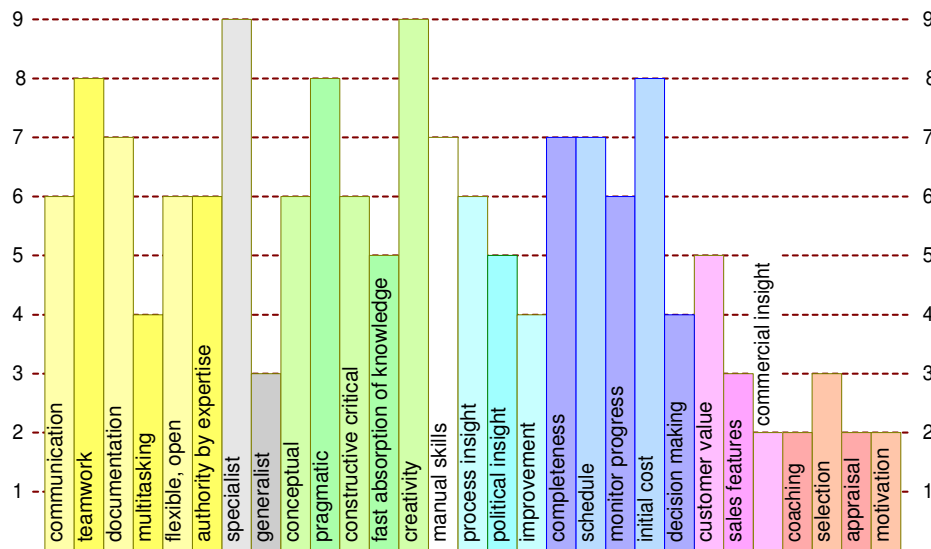


Figure 9.3: The function profile of the developer

9.7 Commercial Manager Profile

The *commercial manager* needs a commercial way of observing and thinking. This way of thinking appears to be fuzzy and not logical for technology oriented people. From technology oriented perspective a strange *mind warp* is required to perform a commercial manager function.

The commercial manager is a valuable complement to the other functions, responsible for aspects such as salability and value proposition.

9.8 Definition of Characteristics

9.8.1 Interpersonal skills

communication The ability to communicate effectively. Communication is a two-way activity, presenting information as well as receiving information is important.

teamwork The ability to work as member of a team, in such a way that the team is more than the collection of individuals.

documentation The ability to create clear, accessible and maintainable documentation in a reasonable amount of time.

multi-tasking The ability to work on many subjects concurrently, where (frequent) external events determine the task switching moments.

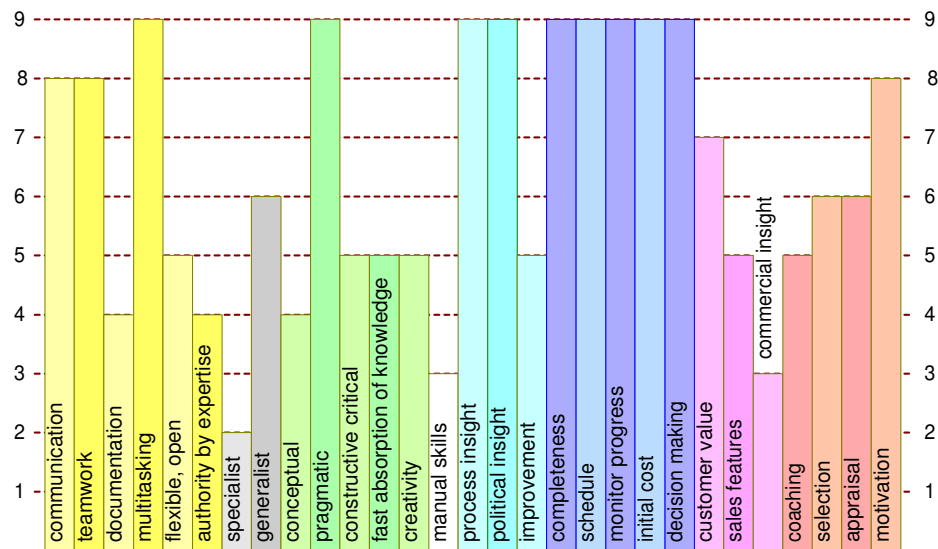


Figure 9.4: The function profile of the operational leader

flexible, open The attitude to respect contributions of others, the willingness to show all personal considerations, even if these are very uncertain, the willingness to adopt solutions of others, even in case of strong personal opinions.

Note that this overall attitude does not mean that a flexible and open person always adopts the ideas of others (chameleon behavior). The true strength of this characteristic is to apply it when necessary, so adopt an alternative solution if it is better.

authority by expertise The personality which convinces people by providing data, instead of citing formal responsibilities. Hard work is required before authority by expertise is obtained; a good track record and trust have to be build up. Authority is earned rather than being enforced.

9.8.2 Know-how

In terms of characteristics the know-how is qualified in 2 categories, generalist and specialist.

Generalist The persons which are always interested in the neighboring areas, how does it fit in the context? How does the “whole” work.

Specialist The persons which are always interested in knowing more detail.

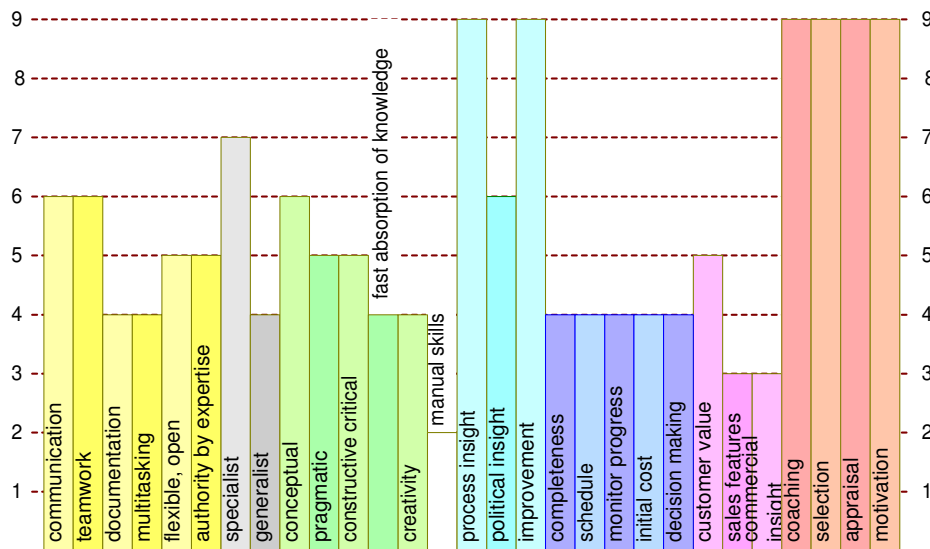


Figure 9.5: The function profile of the line manager

9.8.3 Reasoning Power

conceptual The ability to create the overview, to abstract the concepts from detailed data. The ability to reason in terms of concepts.

pragmatic The ability to accept non-ideal solutions, to go after the 80% solution. The ability to connect "fuzzy" concepts to real world implementations.

constructive critical The ability to identify problems, formulate the problems and to trigger solutions. The term *critical thinking* is also used. Note that critics serves a constructive goal: to achieve better results.

fast absorption of know-how The ability to jump into a new discipline and to absorb the required know-how in a short time. Systems architect are never able to know all about the technologies used in the systems. This capability helps them to get the right knowledge when needed.

creativity The ability to come with new, original ideas. A specific subclass of this ability is lateral thinking: applying know-how from entirely different areas on the problem at hand.

9.8.4 Executing Skills

Manual Skills The ability to **do** things, for instance build or test something. This ability is complementary to the many "mental" skills in this list of character-

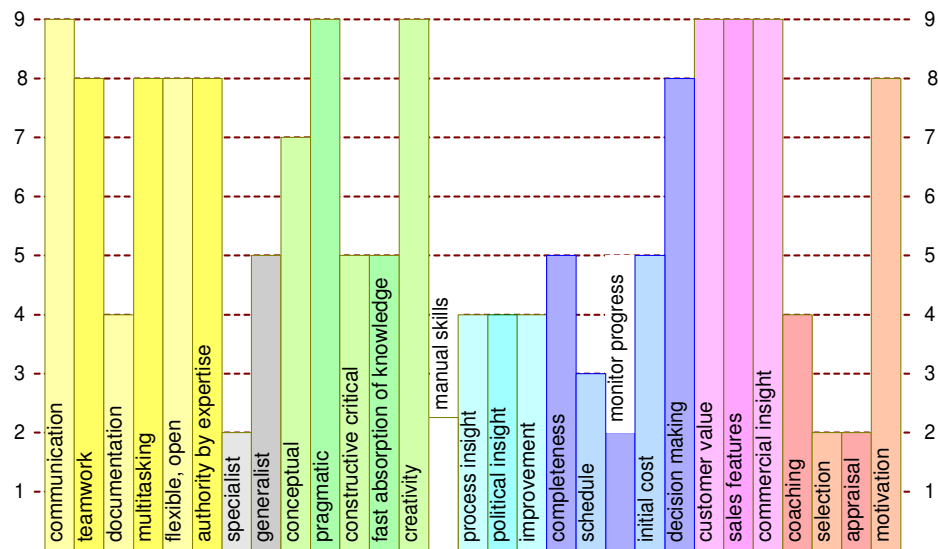


Figure 9.6: The function profile of the commercial manager

istics.

9.8.5 Process Skills

process insight The ability to understand specific processes, the ability to recognize the de facto processes, the ability to assess formal and de facto processes, both the strong points as well as the weak points.

politics insight The ability to recognize the political factors: persons, organizations, motivations, power. The ability to use this information as neutralizing force “depoliticizing”: facts and objectives based decision making instead of power based decision making.

improvement drive The ever present drive to improve the current situation, never getting complacent.

9.8.6 Project Management Skills

Completeness The ability to pursue **all** information. This is often done by means of spreadsheets or databases. Large collections of issues are maintained and processed.

This ability is often complementary to, or even conflicting with, the ability to create understanding and overview: the parts view versus the holistic view.

schedule The ability to create schedules: activities and resources with their relationships, scheduled in time.

monitor progress The ability to monitor progress, the ability to chase people, and the ability to find and resolve the causes of delays.

initial cost The ability to create initial cost estimates and to refine these into budgets. The ability to understand and reason in terms of initial costs. Initial costs are the one time investments needed to develop new products and or businesses.

decision making The ability to make choices and to handle the consequences of these choices.

9.8.7 Commercial Skills

customer value The ability to see and understand the value of a product or service for a customer. The ability to assess the value for the customer.

sales feature The ability to recognize features needed to sell the product. The ability to characterize the relevant characteristics of these features (“tick-mark only”, “competitive edge”, “show-off”, et cetera).

commercial insight The ability to think in commercial terms and concepts, ranging from “branding” to “business models”.

9.8.8 Human Resource Management Skills

coaching The ability to coach other people; help other people by reflection, by stimulating independent thinking and acting.

selection The ability to select individuals for specific jobs. The ability to interview people and to assess them.

appraisal The ability to assess employees and to communicate this assessment in a fair and balanced way.

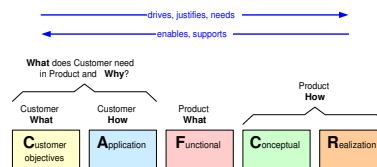
motivation The ability to make people enthusiastic, to motivate them beyond normal performance.

9.9 Acknowledgements

Pierre America applied fine tuning of translations, spelling and capitols. Lennart Hofland suggested an improvement for the description of the commercial manager. Sjir van Loo suggested an increase of coaching and selection skills of the architect. Keith Frampton pointed me to recent research about this subject.

Chapter 10

Short introduction to basic “CAFCR” model



10.1 Introduction

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

An early tutorial[15] of this model used the concatenation of the first letters of the views in this model to form the acronym “CAFCR” (Customer Objectives, Application, Functional, Conceptual, Realization). This acronym is used so often within the company, that changing the acronym has become impossible. We keep the name constant, despite the fact that better names for some of the views have been proposed. The weakest name of the views is *Functional*, because this view also contains the so-called *non functional* requirements. A better name for this view is the Black-Box view, expressing the fact that the system is described from outside, without assumptions about the internals.

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

10.2 The CAFCR model

A useful top level decomposition of an architecture is provided by the so-called “CAFCR” model, as shown in figure 10.1. The *Customer Objectives* view and the *Application* view provide the **why** from the customer. The *Functional* view describes the **what** of the product, which includes (despite the name) also the *non-functional* requirements. The **how** of the product is described in the *Conceptual* and *Realization* view, where the conceptual view is changing less in time than the fast changing realization (Moore’s law!).

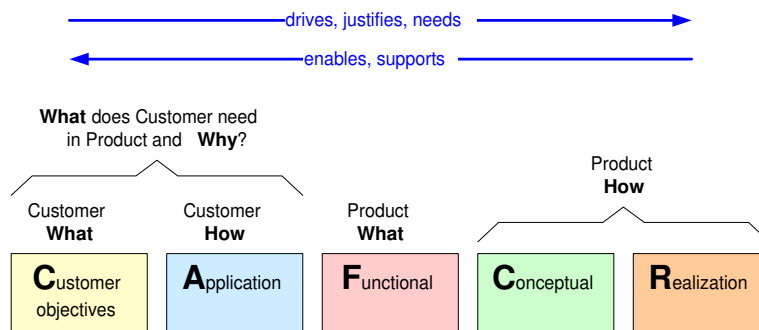


Figure 10.1: The “CAFCR” model

The job of the architect is to integrate these views in a consistent and balanced way. Architects do this job by *frequent viewpoint hopping*: looking at the problem from many different viewpoints, sampling the problem and solution space in order to build up an understanding of the business. Top-down (objective driven, based on intention and context understanding) in combination with bottom-up (constraint aware, identifying opportunities, know how based), see figure 10.2.

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

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

10.3 Who is the customer?

The term *customer* is easily used, but it is far from trivial to determine the customer. The position in the value chain shows that multiple customers are involved. In figure 10.3 the multiple customers are addressed by applying the CAFCR model

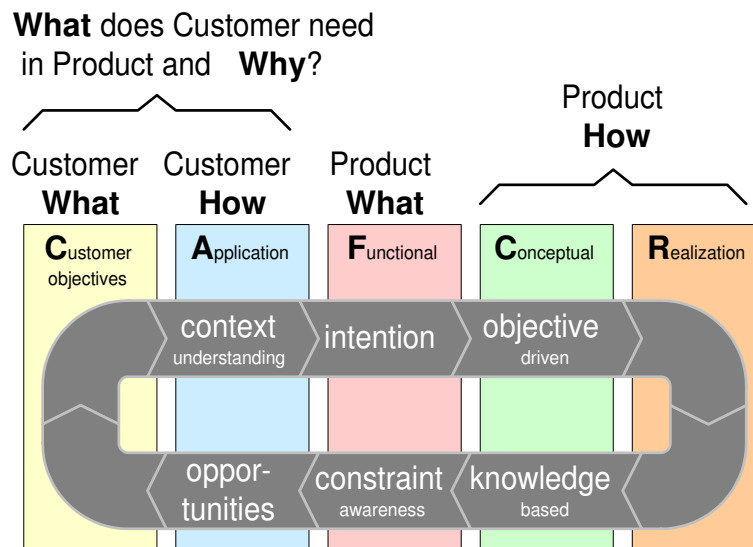


Figure 10.2: Five viewpoints for an architecture. The task of the architect is to integrate all these viewpoints, in order to get a *valuable, usable and feasible* product.

recursively.

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

many stakeholders are involved for any given customer. Multiple stakeholders are involved even when the customer is a consumer, such as parents, other family, and friends. Figure 10.4 shows an example of the people involved in a small company. Note that most of these people have different interests with respect to the system.

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

10.4 Life Cycle view

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

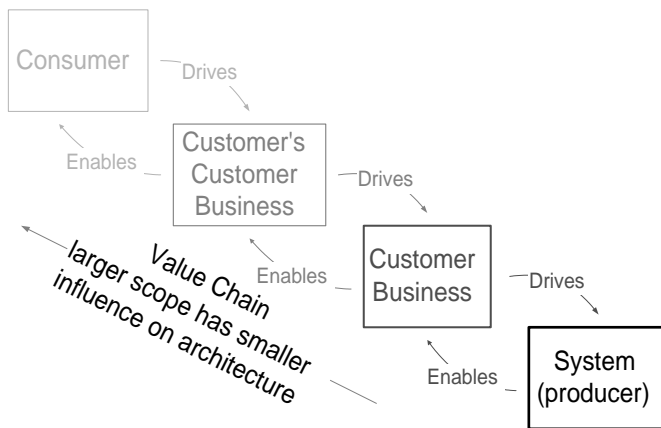


Figure 10.3: CAFCR can be applied recursively

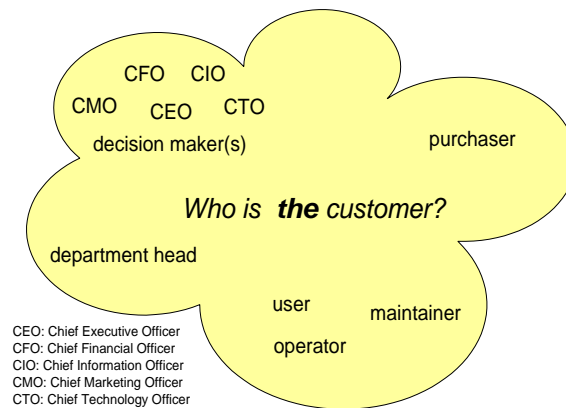


Figure 10.4: Which person is **the** customer?

cycle needs. Figure 10.5 shows the CAFCR+ model that extends the basic CAFCR model with a *Life Cycle view*.

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

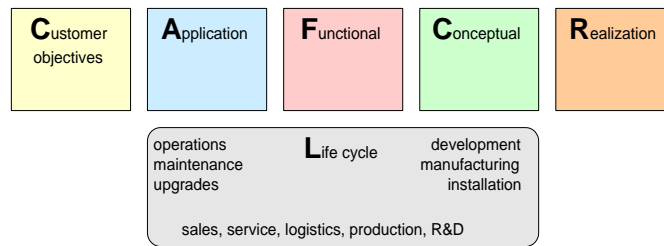


Figure 10.5: CAFCR+ model; Life Cycle View

10.5 Zooming in on security

As an example figure 10.6 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 [9].

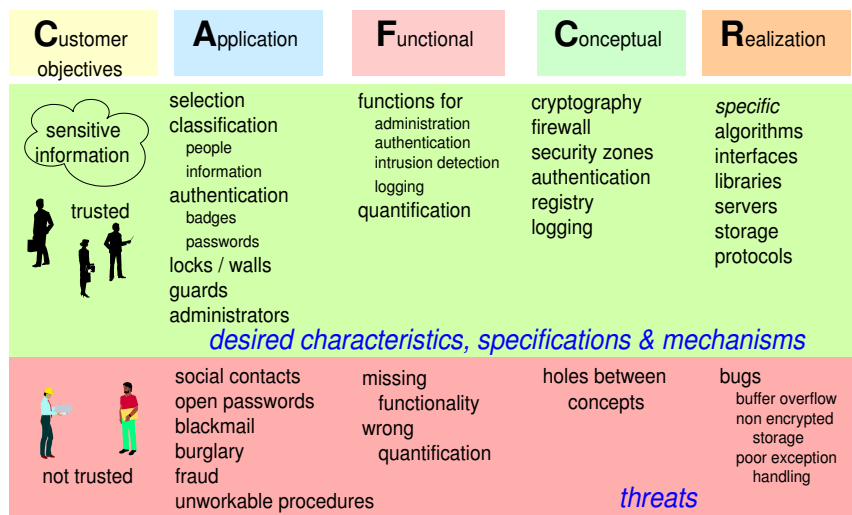


Figure 10.6: Example security through all views

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 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 down the password.

An interesting article is [4], 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.

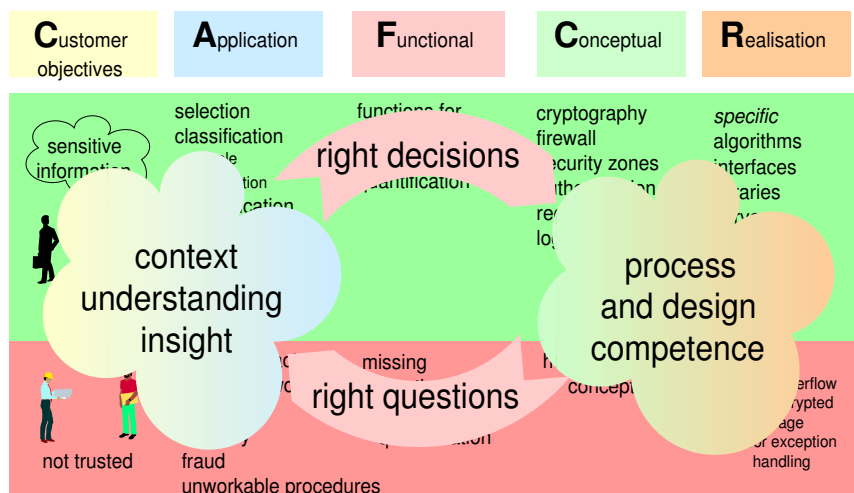


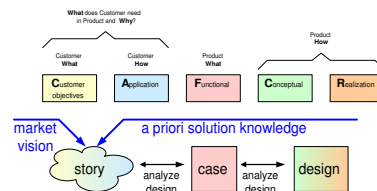
Figure 10.7: Role of views

Figure 10.7 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.

Chapter 11

Story How To



11.1 Introduction

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

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

Figure 11.1 positions the story in the customer objectives view and application view. A good story combines a clear market vision with a priori realization know how. The story itself must be expressed entirely in customer terms, no solution jargon is allowed.

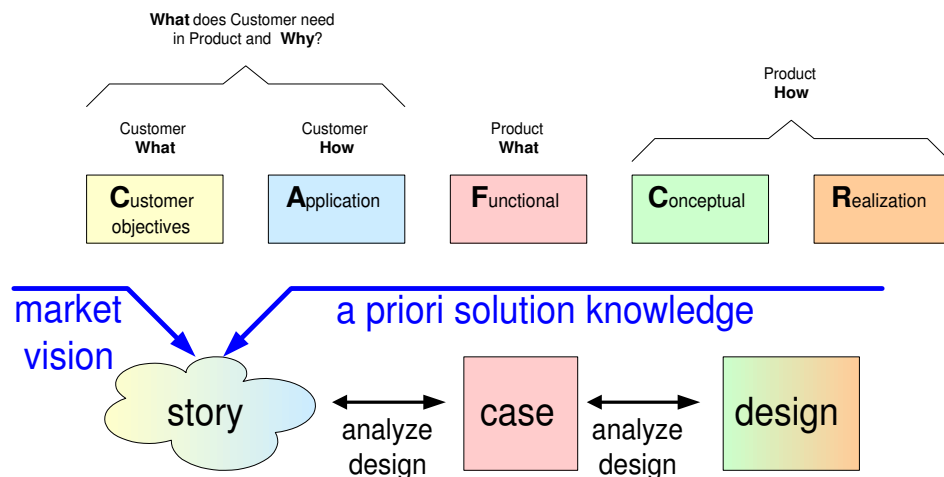


Figure 11.1: From story to design

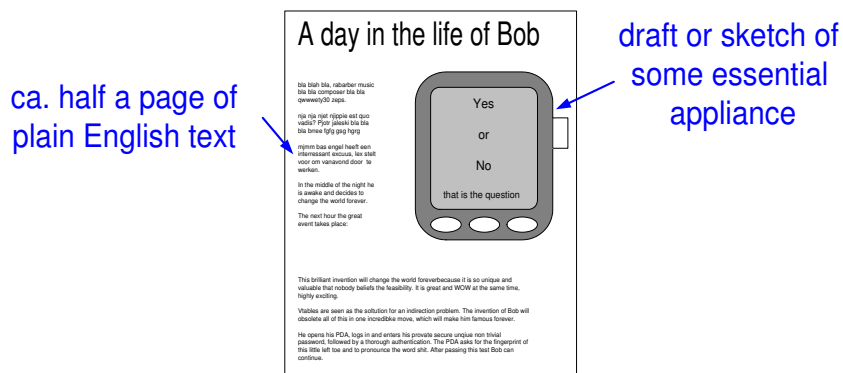


Figure 11.2: Example story layout

11.2 How to Create a Story?

A story is a short single page story, as shown in Figure 11.2, preferably illustrated with sketches of the most relevant elements of the story, for instance the look and feel of the system being used. Other media such as cartoons, animations, video or demonstrations using mockups can be used also. The *duration* or the *size* of the “story” must be limited to enable focus on the essentials.

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

in on aspects of the overall story.

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

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

11.3 How to Use a Story?

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

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

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

11.4 Criteria

Figure 11.3 shows the criteria for a good story. It is recommended to assess a story against this checklist and either improve a story such that it meets all the criteria or to reject the story. Fulfillment of these criteria helps to obtain a useful story. The set of five criteria is a necessary but not sufficient set of criteria. The value of a story can only be measured in retrospect by determining the contribution of the story to the specification and design process.

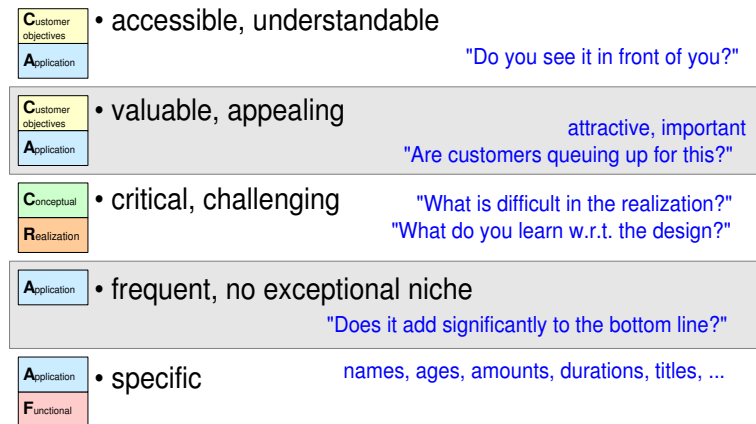


Figure 11.3: criteria for a good story

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

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

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

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

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

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

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

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

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

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

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

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

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

11.5 Example Story

Figure 11.4 shows an example of a story for hearing aids. The story first discusses the problem an elderly lady suffers from due to imperfect hearing aids. The story continues with postulated new devices that helps her to participate again in an active social life.


Figure 11.5 shows for the value and the challenge criteria what this story contributes.

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

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

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

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



source: Roland Mathijssen
Embedded Systems Institute
Eindhoven

Figure 11.4: Example of a story

11.6 Acknowledgements

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

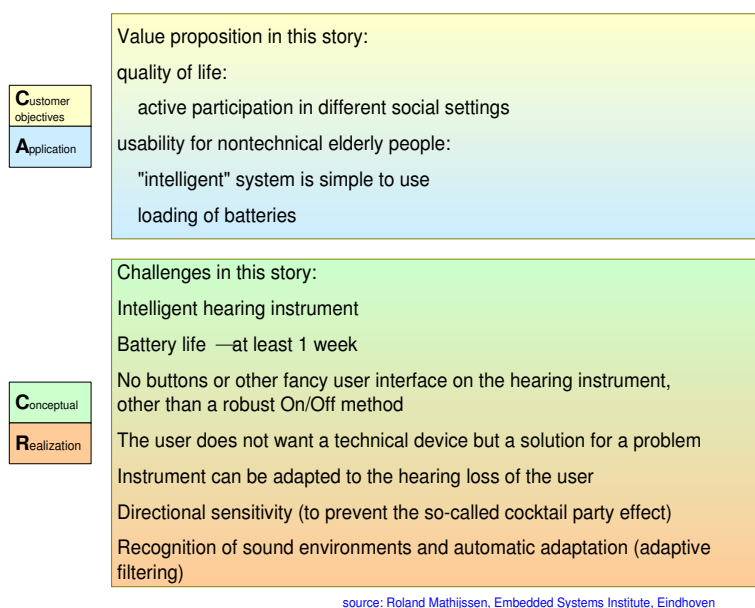


Figure 11.5: Value and Challenges in this story

Bibliography

- [1] Belbin Associates. Belbin team-role summary sheet. http://www.belbin.com/content/page/49/Belbin_Team_Role_Descriptions.pdf, 2001.
- [2] Meredith Belbin. *Management Teams, Why They Succeed or Fail*. Butterworth-Heinemann, Boston, MA, 1981.
- [3] Dana Bredemeyer and Ruth Malan. Role of the software architect. http://www.bredemeyer.com/pdf_files/role.pdf, 1999.
- [4] 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.
- [5] Edward de Bono. Six thinking hats. http://www.debonogroup.com/six_thinking_hats.php.
- [6] DoD. Dod instruction 5000.2. <http://www.dtic.mil/whs/directives/corres/pdf/500002p.pdf>, 2008. see Enclosure 8, page 60-61.
- [7] Nancy Dolan. Human systems integration in dod acquisition. <https://acc.dau.mil/GetAttachment.aspx?id=25755&pname=file&aid=3181&lang=en-US>, unknown.
- [8] K. Frampton, J. M. Carroll, and J. A. Thom. What capabilities do IT architects say they need? In *10th United Kingdom Academy for Information Systems (UKAIS) Proceedings*, 2005.
- [9] 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.

- [10] Gerrit Muller. The system architecture homepage. <http://www.gaudisite.nl/index.html>, 1999.
- [11] Gerrit Muller. From the soft and fuzzy context to SMART engineering. <http://www.gaudisite.nl/FromFuzzyToSmartPaper.pdf>, 2001.
- [12] Gerrit Muller. Architectural reasoning explained. <http://www.gaudisite.nl/ArchitecturalReasoningBook.pdf>, 2002.
- [13] Gerrit Muller and Eirik Hole. Architectural descriptions and models. http://www.architectingforum.org/whitepapers/SAF_WhitePaper_2006_2.pdf. White Paper Resulting from Architecture Forum Meeting March 21-22, 2006 (Washington DC, USA).
- [14] Isabel Myers. *The Myers-Briggs Type Indicator*. Consulting Psychologists Press, Palo Alto, CA, 1962.
- [15] Henk Obbink, Jürgen Müller, Pierre America, and Rob van Ommering. COPA: A component-oriented platform architecting method for families of software-intensive electronic products. http://www.hitech-projects.com/SAE/COPA/COPA_Tutorial.pdf, 2000.
- [16] ONR/SC-21 Manning Affordability Initiative. Human system engineering. [http://www.hf.faa.gov/docs/508/docs/Human_System_Engineering_\(NSWC\).pdf](http://www.hf.faa.gov/docs/508/docs/Human_System_Engineering_(NSWC).pdf).
- [17] Eberhardt Rechtin and Mark W. Maier. *The Art of Systems Architecting*. CRC Press, Boca Raton, Florida, 1997.
- [18] Systems Engineering and Assessment Ltd. The human view; handbook for MODAF. <http://www.hfidtc.com/ModAF/HVHandbookFirstIssue.pdf>, 2008.

History

Version: 0.1, date: February 25, 2009 changed by: Gerrit Muller

- added chapter with background and references to military HFI/HSI work
- added brainstorm in the beginning

Version: 0, date: February 16, 2009 changed by: Gerrit Muller

- Created, no changelog yet