

Reader OOTI course requirements engineering

logo
TBD

Gerrit Muller

University of South-Eastern Norway-NISE

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

gaudisite@gmail.com

Abstract

This book bundles the subjects used in the OOTI course *Requirements Engineering*. All articles are duplicated from other Gaudí articles and books.

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

Introduction	ix
1 Lecture Requirements Engineering	1
1.1 Introduction	1
1.2 Program	1
1.3 Case Description	2
1.4 Instruction for the case	2
1.5 Acknowledgements	4
2 Requirements Capturing by the System Architect	5
2.1 Introduction	5
2.2 Definition of Requirements	5
2.3 System as a black box	7
2.4 Stakeholders	7
2.5 Requirements for Requirements	8
2.6 Viewpoints on Needs	10
2.7 Reference Architecture and Key Drivers	11
2.8 Example Motorway Management	13
2.9 Requirements Value and Selection	13
2.10 Acknowledgements	16
3 The customer objectives view	17
3.1 Introduction	17
3.2 Key drivers	18
3.3 Value chain and business models	20
3.4 Suppliers	21
4 The application view	23
4.1 Introduction	23
4.2 Customer stakeholders and concerns	24
4.3 Context diagram	25
4.4 Entity relationship model	26

4.5	Dynamic models	26
5	Story How To	29
5.1	Introduction	29
5.2	How to Create a Story?	30
5.3	How to Use a Story?	31
5.4	Criteria	31
5.5	Example Story	33
5.6	Acknowledgements	34
6	How to present architecture issues to higher management	36
6.1	Introduction	36
6.2	Preparation	38
6.3	The presentation material	39
6.4	The Presentation	41
6.5	Exercise	43
7	Simplistic Financial Computations for System Architects.	45
7.1	Introduction	45
7.2	Cost and Margin	46
7.3	Refining investments and income	47
7.4	Adding the time dimension	49
7.5	Financial yardsticks	52
7.6	Acknowledgements	54
8	Granularity of Documentation	55
8.1	Introduction	55
8.2	Stakeholders	55
8.2.1	Example digital flat screen TV	56
8.3	Requirements	56
8.4	Documentation Structure	59
8.5	Payload, the ratio between overhead and content	61
8.6	Acknowledgements	62
9	Template How To	63
9.1	Introduction	63
9.2	Why Templates?	64
9.3	New Process Introduction	64
9.4	What does a Template contain	65
9.5	Copy Paste Modify Pattern	66
9.6	Template Development	67
9.7	Guidelines	67
9.8	Pitfalls	69

9.9	Why I hate templates	69
9.10	Summary	70
9.11	Acknowledgements	70

List of Figures

1.1	Schedule of the course	3
1.2	The block diagram of the Cyber Video company system	4
2.1	The flow of requirements	6
2.2	System as a Black Box	7
2.3	A simplified process decomposition of the business. The stakeholders of the requirements are beside the customer self, mainly active in the customer oriented process and the product creation process.	8
2.4	Requirements for Requirements	9
2.5	Complementary viewpoints to collect needs	10
2.6	A Reference Architecture views the architecture from 5 viewpoints	11
2.7	The mapping of Key Drivers via derived application drivers on requirements	12
2.8	Method to define key drivers	12
2.9	Recommendations when defining key drivers	13
2.10	The key drivers, derived application drivers and requirements of a Motorway Management System	14
2.11	The selection process produces a product specification and a phasing and characterization of requirements to prevent repetition of discussion	15
2.12	Simple methods for a first selection	15
2.13	Quantifiable Aspects for Requirements Selection	16
3.1	Overview of Customer Objectives View methods	18
3.2	Example of the four key drivers in a motorway management system	19
3.3	Submethod to link key drivers to requirements, existing of the iteration over four steps	20
3.4	Recommendations for applying the key driver submethod	20
3.5	Example value chain	21
3.6	Example of simple supplier map for a cable provider	22

4.1	Overview of methods and models that can be used in the application view	24
4.2	Stakeholders and concerns of an MRI scanner	25
4.3	Systems in the context of a motorway management system	25
4.4	Diagram with entities and relationship for a simple TV appliance	26
4.5	Examples of dynamic models	27
4.6	Productivity and cost models	28
4.7	Dynamics of an URF examination room	28
5.1	From story to design	30
5.2	Example story layout	30
5.3	criteria for a good story	32
5.4	Example of a story	34
5.5	Value and Challenges in this story	35
6.1	Architectural issues related to managerial viewpoints	37
6.2	Characteristics of managers in higher management teams	37
6.3	How to prepare	38
6.4	Recommended content	39
6.5	Mentioned info, shown info and backup info	40
6.6	Form is important	40
6.7	Don't force your opinion, understand the audience	41
6.8	How to cope with managerial dominance	42
6.9	Exercise presentation to higher management	43
6.10	Schedule of the presentation exercise	44
7.1	The relation between sales price, cost price and margin per product	46
7.2	Profit as function of sales volume	47
7.3	Investments, more than R&D	48
7.4	Income, more than product sales only	49
7.5	The Time Dimension	50
7.6	The "Hockey" Stick	50
7.7	What if ...?	51
7.8	Stacking Multiple Developments	52
7.9	Fashionable financial yardsticks	53
8.1	The stakeholders of a single document	56
8.2	Large documents are decomposed in smaller documents, supported by a document structure and overview	59
8.3	Decomposition is applied recursively until the atomic documents fulfill the requirements in section 8.3	60
8.4	Layout of a good document, heuristic for the number of pages of a good document is $4 \leq nrofpages \leq 20$	61

9.1	The reactionary force induced by the proposed new process is countered by giving support	64
9.2	The relation between a template and a process	65
9.3	Templates from layout only, up to prescribing structure or contents, templates to support meta information are recommended.	65
9.4	Spiral development model for Templates	67

List of Tables

9.1	<i>Overview of Template characteristics</i>	66
-----	---	----

Introduction

This book bundles the articles produced by the Gaudí project.

At this moment the book is in its early infancy. Most articles are updated based on feedback from readers and students. The most up to date version of the articles can always be found at [12]. The same information can be found here in presentation format.

Chapters can be read as autonomous units. The sequence chosen here is more or less top down, hopping from one viewpoint to the next. On a regular base a sidestep ("Intermezzo") is being made, either to describe a more fundamental notion, or to propose a more challenging point of view.

Chapter 1

Lecture Requirements Engineering

block 1: teacher provides case		
What are requirements, black box, SMART		1/2 day
homework: make requirement specification		
Customer and Application view, Story telling		1/2 day
homework: improve requirement specification		
Discussion of requirement specification per team		1/2 day

block 2: actual current case of OOTI education		
Financial viewpoint, Presentation to management		1/2 day
homework: make presentation outline		
Documentation How-to	Coaching and discussion	1/2 day
homework: make presentation		
Presentation project case to management team		1/2 day
individual report		

1.1 Introduction

This article describes the Requirements Engineering session part of the Software Engineering block in the OOTI curriculum of the Technical University Eindhoven. Trainer is the author of this article Gerrit Muller.

The focus of this course is on capturing and managing requirements. The notion of key drivers will be introduced as a means to capture and manage. A case is used as learning vehicle. The students have to perform the requirements analysis in this case. The findings of the requirements analysis have to be presented to a management team and then to be written down in a requirement specification.

1.2 Program

The lecture program is:

time	subject
Session 1	What are requirements, black box, SMART
Session 2	Customer and Application view, Story telling
Session 3	Discussion of requirement specification per team
Session 4	Financial viewpoint, Presentation to management
Session 5	Documentation How-to, Coaching and discussion session
Session 6	Presentation of project case to management team

The time in between lectures is to be used to perform a case study. The case study will be explained on the first half day. Half a day must be used to explore the case, During the next half lecture day the status of the case will be discussed and clarifications will be given. At the end of the block the case should be finished and the results will be presented and discussed. The course is closed by writing a summary of the case findings (per group) and lessons learned per individual, see section 1.4. Figure 1.1 shows the schedule of the course on a timeline.

1.3 Case Description

A video content distribution company is planning to deliver video which can be transferred to a local box in the house of the consumer via satellite. Figure 1.2 shows a diagram of the system.

1.4 Instruction for the case

The case is performed in 4 groups of 3..5 people, working together on the same problem. Instructions for the case:

1. Block 1 session 1: Make an initial requirements specification
2. Block 1 session 2: Improve and complete requirements specification
3. Block 2 session 4: Make an outline of a presentation of maximum 10 minutes, target audience: management team of your company
4. Block 2 session 5: Prepare and exercise presentation
5. Block 2 session 6: Write an individual report reflecting on: requirement specification, management presentation, lessons learned and how to do it next time.

Recommended way of working:

1. Make a black box view of the system

block 1; teacher provides case		
What are requirements, black box, SMART		1/2 day
homework: make requirement specification		
Customer and Application view, Story telling		1/2 day
homework: improve requirement specification		
Discussion of requirement specification per team		1/2 day
block 2; actual current case of OOTI education		
Financial viewpoint, Presentation to management		1/2 day
homework: make presentation outline		
Documentation How-to	Coaching and discussion	1/2 day
homework: make presentation		
Presentation project case to management team		1/2 day
individual report		

Figure 1.1: Schedule of the course

2. Make some initial drafts and designs to explore the problem.
3. Make a story which helps to understand the products, make sure to use the criteria for a story.
4. Look from all stakeholder points of view towards the problem and identify what they need and what they expect.
5. Analyze the information obtained so far and extract the underlying requirements.
6. Abstract the key drivers behind the requirements.
7. Make a top-down description of the requirements.

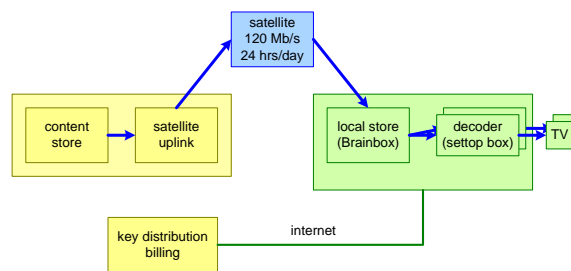


Figure 1.2: The block diagram of the Cyber Video company system

Every group will present its findings on day 5, followed by a short discussion. This presentation is to the management team of the company which will make these products and some invited lead customers.

Create a *Requirement specification* which can be used as the starting point of the design. On day 6 a coaching session is held to discuss the results so far, on day 7 the requirement specifications are reviewed.

Write an individual summary of the entire process, maximum 2 A4's, touching the following questions:

- What are the most important lessons you learned from these exercise (requirement specification, management presentation)?
- Which roles did the members of the group play during the exercise?
- How would you approach such a problem the next time?
- Which stakeholders understand your group presentation? Are they happy with the presentation?

Don't answer all these questions perfectly, finish the summary in at most half a day.

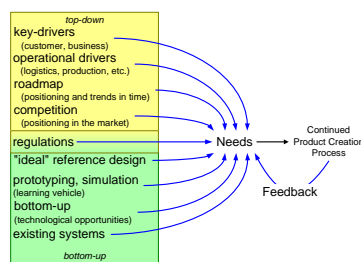
1.5 Acknowledgements

The case used in this course is derived from the case defined by Sjjir van Loo for use in the course SW architecture. The case defined by Sjjir is further reduced in this course to stimulate the students in the requirements exploration, reflecting real-life situations which often start rather ill-defined.

I thank Sjjir van Loo for providing his course material. I also thank Dieter Hammer and Harold Weffers for the initial discussions and for the suggestion to use this case.

Chapter 2

Requirements Capturing by the System Architect



2.1 Introduction

The basis of a good system architecture is the availability and understanding of the needs of all stakeholders. Stakeholder needs are primary inputs for the system specification. The terms *requirements elicitation*, *requirements analysis*, and *requirements management* are frequently used as parts of the Product Creation Process that cope with the transformation of needs into specification and design.

2.2 Definition of Requirements

The term requirement is quite heavily overloaded in Product Creation context. *Requirement* is sometimes used non-obligatory, e.g. to express wants or needs. In other cases it is used as mandatory prescription, e.g. a must that will be verified. Obviously, dangerous misunderstandings can grow if some stakeholders interpret a requirement as want, while other stakeholders see it as must.

We will adopt the following terms to avoid this misunderstanding:

Customer Needs The term *Customer Needs* is used for the non-mandatory wishes, wants, and needs.

Product Specification The term *Product Specification* is used for the mandatory characteristics the system must fulfill.

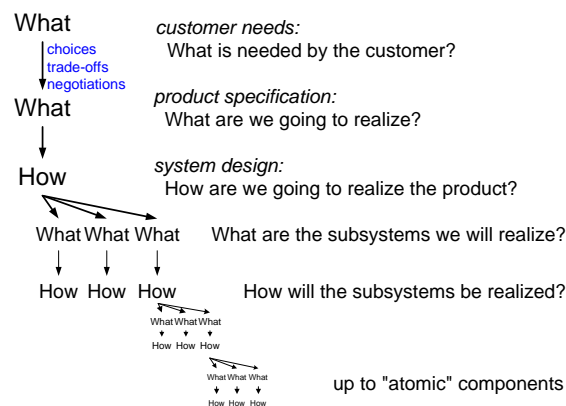


Figure 2.1: The flow of requirements

In the system engineering world the term *Requirements Management* or *Requirements Engineering* is also being used. This term goes beyond the two previous interpretations. The requirements management or engineering process deals with the propagation of the requirements in the product specification towards the requirements of the atomic components. Several propagation steps take place between the product specification and atomic components, such as requirements of the subsystems defined by the first design decomposition. In fact the requirement definition is recursively applied for every decomposition level similar to the product specification and subsystem decomposition.

Figure 2.1 shows the requirements engineering flow. The customer needs are used to determine the product specification. Many choices are made going from needs to specification, sometimes by negotiation, sometimes as trade-off. Often the needs are not fully satisfied for mundane reasons such as cost or other constraints. In some cases the product specification exceeds the formulated needs, for instance anticipating future changes.

Figure 2.1 also show the separation of specification, *what*, and design, *how*. This separation facilitates clear and sharp decision making, where goals *what* and means *how* are separated. In practice decision are often polluted by confusing goals and means.

An other source of requirements is the organization that creates and supplies the product. The needs of the organization itself and of the supply and support chain during the life cycle are described in this chapter as *Life Cycle Needs*.

2.3 System as a black box

One of the main characteristics of requirements in the product specification is that they describe *what* has to be achieved and not *how* this has to be achieved. In other words, the product specification describes the system as *black box*. Figure 2.2 provides a starting point to write a product specification.

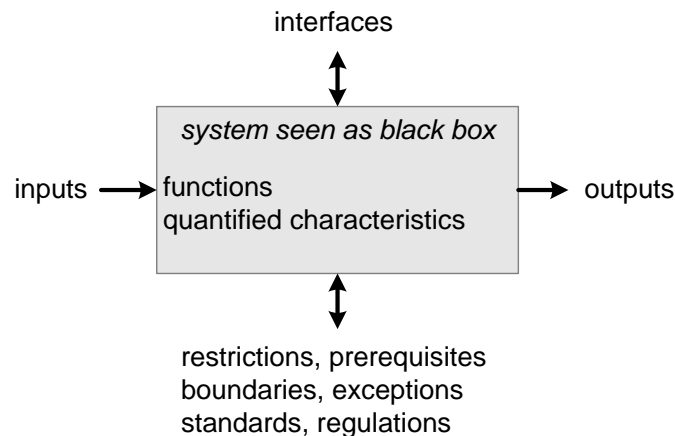


Figure 2.2: System as a Black Box

The system is seen as black box. What goes into the box, what comes out and what functions have to be performed on the inputs to get the outputs. Note that the functions tell something about the black box, but without prescribing how to realize them. All interfaces need to be described, interfaces between the system and humans as well as interfaces to other systems. The specification must also quantify desired characteristics, such as how fast, how much, how large, how costly, et cetera.

Prerequisites and constraints enforced on the system form another class of information in the product specification. Further scoping can be done by stating boundaries and desired behavior in case of exceptions. Regulations and standards can be mandatory for a system, in which case compliance with these regulations and standards is part of the product specification.

2.4 Stakeholders

A simplified process model is shown in figure 2.3. The stakeholders of the product specification are of course the customers, but also people in the Customer Oriented Process, the Product Creation Process, People, Process, and Technology Management

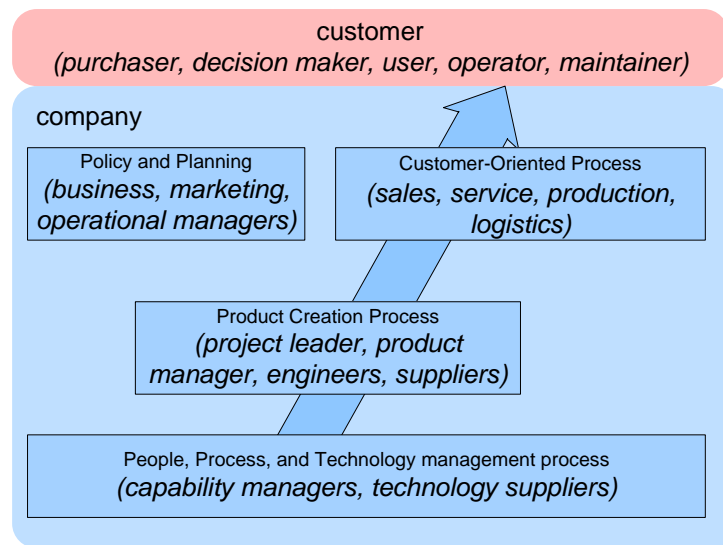


Figure 2.3: A simplified process decomposition of the business. The stakeholders of the requirements are beside the customer self, mainly active in the customer oriented process and the product creation process.

Process, and the Policy and Planning Process. The figure gives a number of examples of stakeholders per process.

The customer can be a consumer, but it can also be a business or even a group of businesses. Businesses are complex entities with lots of stakeholders. A good understanding of the customer business is required to identify the customer-stakeholders.

2.5 Requirements for Requirements

Standards like ISO 9000 or methods like CMM prescribe the requirements for the requirements management process. The left side of Figure 2.4 shows typical requirements for the requirements itself.

Specific , what is exactly needed? For example, The system shall be *user friendly* is way too generic. Instead a set of specific requirements is needed that together will contribute to user friendliness.

Unambiguous so that stakeholders don't have different expectations on the outcome. In natural language statements are quite often context sensitive, making the statement ambiguous.

Verifiable so that the specification can be verified when realized.

Specific	
Unambiguous	
Verifiable	Accessible
Quantifiable	Understandable
Measurable	Low threshold
Complete	
Traceable	

Figure 2.4: Requirements for Requirements

Quantifiable is often the way to make requirements verifiable. Quantified requirements also help to make requirements specific

Measurable to support the verification. Note that not all quantified characteristics can also be measured. For example in wafer steppers and electron microscopes many key performance parameters are defined in nanometers or smaller. There are many physical uncertainties to measure such small quantities.

Complete for all main requirements. *Completeness* is a dangerous criterion. In practice a specification is never complete, it would take infinite time to approach completeness. The real need is that all crucial requirements are specified.

Traceable for all main relations and dependencies. *Traceability* is also a dangerous criterion. Full traceability requires more than infinite time and effort. Understanding how system characteristics contribute to an aggregate performance supports reasoning about changes and decision making.

Unfortunately, these requirements are always biased towards the formal side. A process that fulfills these requirements is from theoretical point of view sound and robust. However, an aspect that is forgotten quite often, is that product creation is a human activity, with human capabilities and constraints. The human point of view adds a number of requirements, shown at the right hand side of Figure 2.4: accessibility, understandability, and a low threshold. These requirements are required for **every** (human) stakeholder.

These requirements, imposed because of the human element, can be conflicting with the requirements prescribed by the management process. Many problems in practice can be traced back to violation of the human imposed requirements. For instance, an abstract description of a customer requirement such that no real customer can understand the requirements anymore. Lack of understanding is a severe risk, because early validation does not take place.

2.6 Viewpoints on Needs

Needs for a new product can be found in a wide variety of sources. The challenge in identifying needs is, in general, to distinguish a solution for a need from the need itself. Stakeholders, when asked for needs, nearly always answer in terms of a solution. For example, consumers might ask for a *flash based video recorder*, where the underlying need might be a light-weight, small, portable video recorder. It is the architect's job, together with marketing and product managers, to reconstruct the actual needs from the answers that stakeholders give.

Many complementary viewpoints provide a good collection of needs. Figure 2.5 shows a useful number of viewpoints when collecting needs.

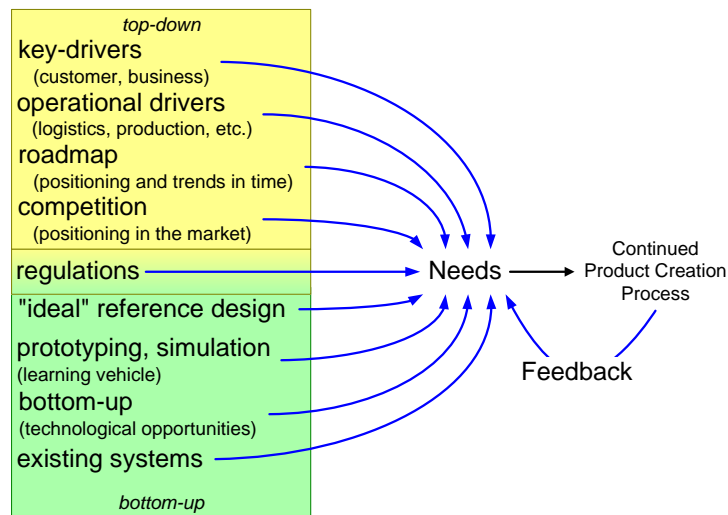


Figure 2.5: Complementary viewpoints to collect needs

The **key-driver** viewpoint and the **operational** viewpoint are the viewpoints of the stakeholders who are “consuming” or “using” the output of the Product Creation Process. These viewpoints represent the “demand side”.

The **roadmap** and the **competition** viewpoints are viewpoints to position the products in time and in the market. These viewpoints are important because they emphasize the fact that a product is being created in a dynamic and evolving world. The product context is not static and isolated.

Regulations result in requirements both top-down, as well as bottom-up. A top down example are labor regulations that can have impact on product functionality and performance. A bottom up example are materials regulations, for instance do not use lead, that may strongly influence design options.

The **“ideal” reference design** is the challenge for the architect. What is in the architect’s vision the perfect solution? From this perfect solution the implicit needs

can be reconstructed and added to the collection of needs.

Prototyping or simulations are an important means in communication with customers. This “pro-active feedback” is a very effective filter for nice but impractical features at the one hand and it often uncovers many new requirements. An approach using only concepts easily misses practical constraints and opportunities.

The **bottom up** viewpoint is the viewpoint where the technology is taken as the starting point. This viewpoint sometimes triggers new opportunities that have been overlooked by the other viewpoints due to an implicit bias towards today’s technology.

The **existing system** is one of the most important sources of needs. In fact it contains the accumulated wisdom of years of practical application. Especially a large amount of small, but practical, needs can be extracted from existing systems.

The product specification is a dynamic entity, because the world is dynamic: the users change, the competition changes, the technology changes, the company itself changes. For that reason the **Continuation of the Product Creation Process** will generate input for the specification as well. In fact nearly all viewpoints are present and relevant during the entire Product Creation Process.

2.7 Reference Architecture and Key Drivers

A system architect must look at the product from multiple complementary viewpoints. Figure 2.6 shows 5 useful views for a reference architecture.

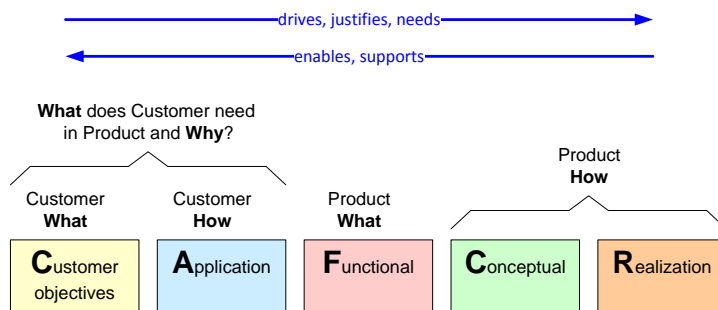


Figure 2.6: A Reference Architecture views the architecture from 5 viewpoints

The business architecture is the architecture of the business of the customer, in relation with the product. Typically it will describe the flow of information or goods, the business processes and the related roles.

A very powerful means to capture requirements is to describe the essence of the business in terms of *Key Drivers*. These drivers must be recognized and understood by the customer, which means that these drivers should be expressed in the language of the customer. A maximum of 5 Key Drivers is recommended to

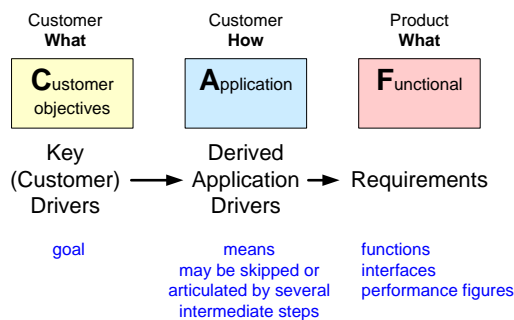


Figure 2.7: The mapping of Key Drivers via derived application drivers on requirements

maintain focus on the essence, the name is on purpose **Key** driver. The key drivers are one aspect of the business architecture. Figure 3.3 shows a method to define key drivers. Figure 3.4 shows some recommendations with respect to the definition of key drivers.

• Define the scope specific.	in terms of stakeholder or market segments
• Acquire and analyze facts	extract facts from the product specification and ask why questions about the specification of existing products.
• Build a graph of relations between drivers and requirements by means of brainstorming and discussions	where requirements may have multiple drivers
• Obtain feedback	discuss with customers, observe their reactions
• Iterate many times	increased understanding often triggers the move of issues from driver to requirement or vice versa and rephrasing

Figure 2.8: Method to define key drivers

Key drivers can be mapped on derived application drivers. Which application activities are done to enable the key driver? The derived application drivers must also be expressed in customer language. The explicit description of application drivers will also ease the job of modelling the application domain.

The derived application drivers are implemented or supported by features or functions of the product. This means that the derived application drivers can be translated into customer requirements of the product.

From point of view of requirements engineering the customer requirements are used as input to produce a product specification, which controls the entire product creation process. The design of the system will result in a technical architecture, with amongst others a decomposition in subsystems and function allocation. The technical architecture is finally mapped onto an implementation. The relation between requirements at the functional architecture level, the technical architecture level and the implementation is managed by the requirements management process.

• Limit the number of key-drivers	minimal 3, maximal 6
• Don't leave out the obvious key-drivers	for instance the well-known main function of the product
• Use short names, recognized by the customer.	
• Use market-/customer- specific names, no generic names	for instance replace "ease of use" by "minimal number of actions for experienced users", or "efficiency" by "integral cost per patient"
• Do not worry about the exact boundary between Customer Objective and Application	create clear goal means relations

Figure 2.9: Recommendations when defining key drivers

Approaching the requirements definition in this way enables the architect to understand a technical feature in relation with the key driver from the customer business. Any feature that cannot be related back to a key driver is suspect: either it should not be there or some requirement or driver is missing.

2.8 Example Motorway Management

Figure 3.2 shows an example of the requirements analysis of a motorway management system. The keydrivers of a motorway management owner are:

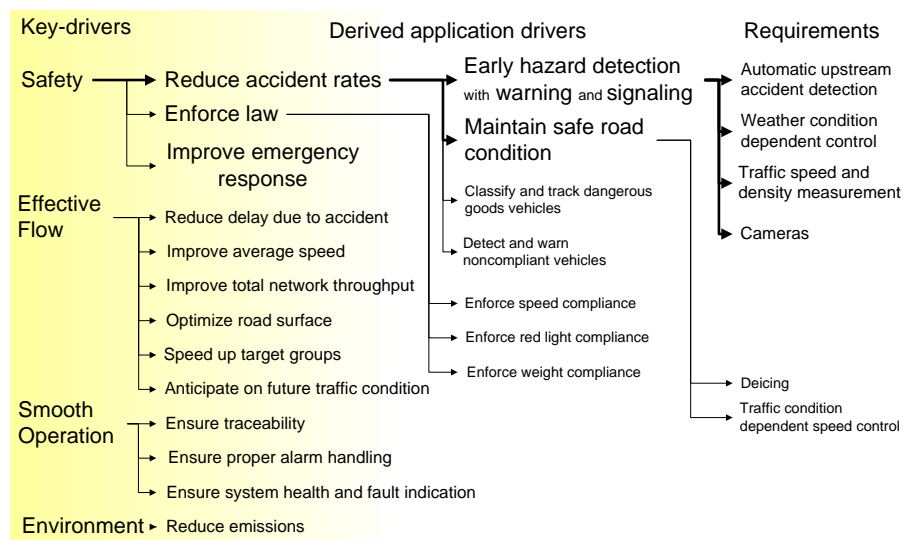
- Safety
- Effective Flow
- Smooth Operation
- Environment

To realize these key drivers the owner applies a number of application processes. This leads to the derived application drivers. For instance to realize safety it is important to prevent accidents and to have immediate response by emergency departments in case of accidents.

2.9 Requirements Value and Selection

The collection of customer and operational needs is often larger than can be realized in the first release of a product. A selection step is required to generate a product specification with the customer and operational needs as input. Figure 2.11 shows the selection process as black box with its inputs and outputs.

The selection process is primarily controlled by the strategy of the company. The strategy determines market, geography, timing and investments. The roadmap, based on the strategy, is giving context to the selection process for a individual



Note: the graph is only partially elaborated for application drivers and requirements

Figure 2.10: The key drivers, derived application drivers and requirements of a Motorway Management System

products. The reality of the competitive market is the last influencing factor on the selection.

The selection will often be constrained by technology, people, and process. The decisions in the selection require facts or estimates of these constraints.

During the selection a lot of insight is obtained in needs, the value of needs, and the urgency. We recommend to consolidate these insights, for example by documenting the characterization of needs. The timing insights can be documented in a phased plan for requirements.

The amount of needs can be so large that it is beneficial to quickly filter out the “obvious” requirements. For some needs it is immediately obvious that they have to be fulfilled anyway, while other needs can be delayed without any problem. Figure 2.12 shows a number of qualitative characterizations of needs, visualized in a two-dimensional matrix. For every quadrant in the matrix a conclusion is given, a need must be included in the specification, a need has to be discarded or the need must be discussed further.

This simple qualitative approach can, for instance, be done with the following criteria:

- importance versus urgency

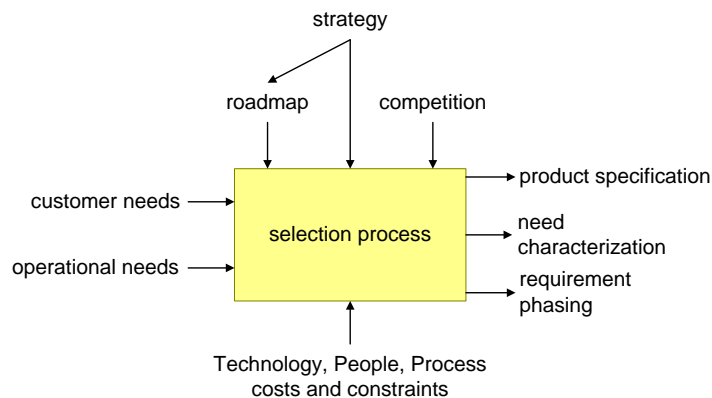


Figure 2.11: The selection process produces a product specification and a phasing and characterization of requirements to prevent repetition of discussion

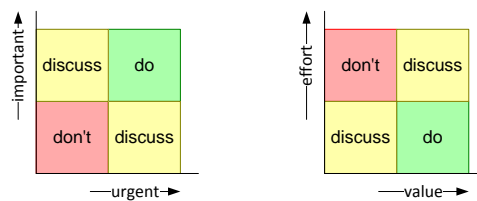


Figure 2.12: Simple methods for a first selection

- customer value versus effort

In the final selection step a more detailed analysis step is preferable, because this improves the understanding of the needs and results in a less changes during the development.

A possible way to do this more detailed analysis is to “quantify” the characteristics for every requirement for the most business relevant aspects, see for examples Figure 2.13.

These quantifications can be given for the immediate future, but also for the somewhat remote future. In that way insight is obtained in the trend, while this information is also very useful for a discussion on the timing of the different requirements. In [3] a much more elaborated method for requirement evaluation and selection is described.

The output of the requirement characterization and the proposed phasing can be used as input for the next update cycle of the roadmap.

<ul style="list-style-type: none"> • Value for the customer • (dis)satisfaction level for the customer • Selling value (How much is the customer willing to pay?) • Level of differentiation w.r.t. the competition • Impact on the market share • Impact on the profit margin
Use relative scale, e.g. 1..5 1=low value, 5 -high value
Ask several knowledgeable people to score
Discussion provides insight (don't fall in spreadsheet trap)

Figure 2.13: Quantifiable Aspects for Requirements Selection

2.10 Acknowledgements

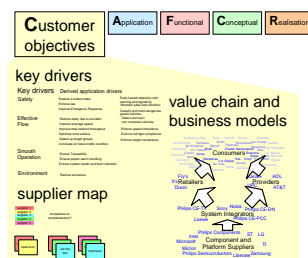
The platform project within Philips Projects provided a clear analysis of amongst others a motorway management process. Wil Hoogenstraaten contributed significantly in the creation of this requirements analysis and sharpened the model from key driver towards features and functions.

Stimulating discussions with Henk Obbink and Jürgen Müller helped to shape this article.

Shakil Ahmed added the regulations viewpoint.

Chapter 3

The customer objectives view



3.1 Introduction

The customer objectives view describes the goals of the customer, the **what**. The goal of articulating these objectives is to better understand the needs and therefore to be able to design a better product.

In searching the objectives some focus on the product is needed, although the architect must keep an open mind. The architect must prevent a circular reasoning, starting from the product functionality and, blinded by the product focus, finding only objectives matching with this same functionality.

Ideally the trade-offs in the customer domain become clear. For instance what is the trade-off between performance and cost, or size and performance or size and cost. The key driver method articulates the essence of the customer needs in a limited set of drivers.

The customer is often driven by his context. Some of the models and methods described here address ways to understand the customer context, such as value chains and business models. Value chains and business models are used to address the customer's customer. The supplier map addresses the supplying side of the customer.

Figure 3.1 shows an overview of the methods in the customer objectives view.

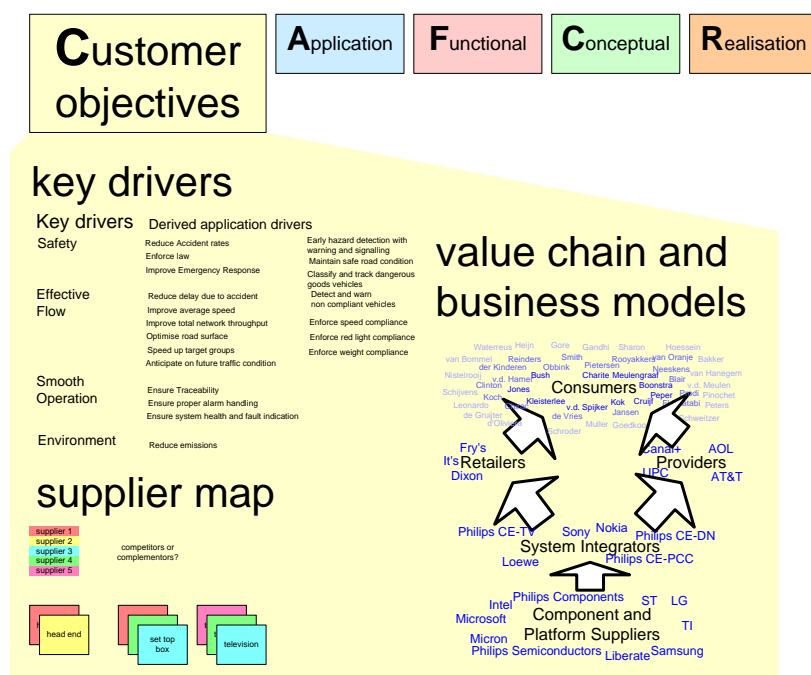


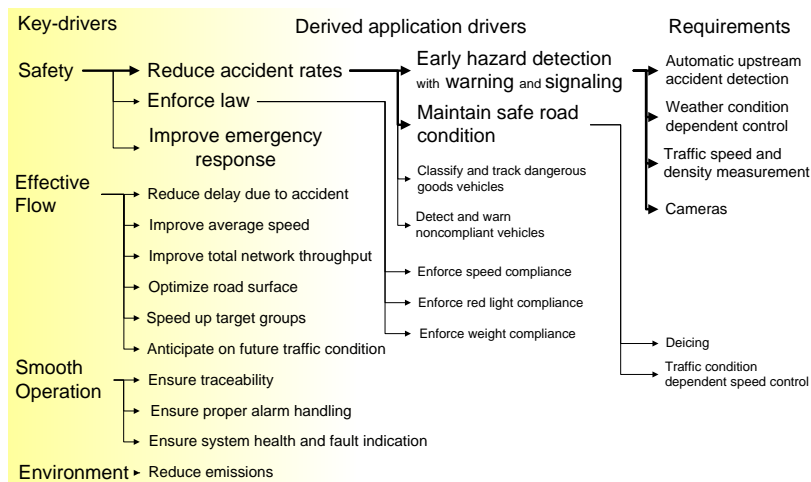
Figure 3.1: Overview of Customer Objectives View methods

3.2 Key drivers

The essence of the objectives of the customers can be captured in terms of customer key drivers. The key drivers provide direction to capture requirements and to focus the development. The key drivers in the customer objectives view will be linked with requirements and design choices in the other views. The key driver submethod gains its value from relating a few sharp articulated key drivers to a much longer list of requirements. By capturing these relations a much better understanding of customer and product requirements is achieved.

Figure 3.2 shows an example of key drivers for a motorway management system, an analysis performed at Philips Projects in 1999.

Figure 3.3 shows a submethod how to obtain a graph linking key drivers to requirements. The first step is to define the scope of the key driver graph. For Figure 3.2 the customer is the motorway management operator. The next step is to acquire facts, for example by extracting functionality and performance figures out of the product specification. Analysis of these facts recovers implicit facts. The requirements of an existing system can be analyzed by repeating *why* questions. For example: “Why does the system need *automatic upstream accident detection*?”. The third step is to bring more structure in the facts, by building a graph, which



Note: the graph is only partially elaborated for application drivers and requirements

Figure 3.2: Example of the four key drivers in a motorway management system

connects requirements to key drivers. A workshop with brainstorming and discussions is an effective way to obtain the graph. The last step is to obtain feedback from customers. The total graph can have many n:m relations, i.e. requirements that serve many drivers and drivers that are supported by many requirements. The graph is good if the customers are enthusiastic about the key drivers and the derived application drivers. If a lot of explaining is required then the understanding of the customer is far from complete. Frequent iterations over these steps improve the quality of the understanding of the customer's viewpoint. Every iteration causes moves of elements in the graph in driver or requirement direction and also causes rephrasing of elements in the graph.

Figure 3.4 shows an additional set of recommendations for applying the key driver submethod. The most important goals of the customer are obtained by limiting the number of key drivers. In this way the participants in the discussion are forced to make choices. The focus in product innovation is often on differentiating features, or unique selling points. As a consequence, the core functionality from the customer's point of view may get insufficient attention. An example of this are cell phones that are overloaded with features, but that have a poor user interface to make connections. The core functionality must be dominantly present in the graph. The naming used in the graph must fit in the customer world and be as specific as possible. Very generic names tend to be true, but they do not help to really understand the customer's viewpoint. The boundary between the Customer Objectives view and the Application view is not very sharp. When creating the

• Define the scope specific.	in terms of stakeholder or market segments
• Acquire and analyze facts	extract facts from the product specification and ask why questions about the specification of existing products.
• Build a graph of relations between drivers and requirements by means of brainstorming and discussions	where requirements may have multiple drivers
• Obtain feedback	discuss with customers, observe their reactions
• Iterate many times	increased understanding often triggers the move of issues from driver to requirement or vice versa and rephrasing

Figure 3.3: Submethod to link key drivers to requirements, existing of the iteration over four steps

• Limit the number of key-drivers	minimal 3, maximal 6
• Don't leave out the obvious key-drivers	for instance the well-known main function of the product
• Use short names, recognized by the customer.	
• Use market-/customer- specific names, no generic names	for instance replace "ease of use" by "minimal number of actions for experienced users", or "efficiency" by "integral cost per patient"
• Do not worry about the exact boundary between Customer Objective and Application	create clear goal means relations

Figure 3.4: Recommendations for applying the key driver submethod

graph that relates *key drivers* to *requirements* one frequently experiences that a key driver is phrased in terms of a (partial) solution. If this happens either the key driver has to be rephrased or the solution should be moved to the requirement (or even realization) side of the graph. A repetition of this kind of iterations increases the insight in the needs of the customer in relation to the characteristics of the product. The **why**, **what** and **how** questions can help to rephrase drivers and requirements. The graph is good if the relations between goals and means are clear for all stakeholders.

3.3 Value chain and business models

The position of the customer in the value chain and the business models deployed by the players in the value chain are important factors in understanding the goals of this customer.

Figure 3.5 shows an example value chain from the Consumer Electronics Domain. At the start of the chain are the component suppliers, making chips and other elementary components such as optical drives, displays, et cetera. These compo-

nents are used by system integrators, building the consumer appliances, such as televisions, set top boxes and cellphones. Note that this value chain is often longer than shown here, where components are aggregated in larger components into subassemblies and finally into systems.

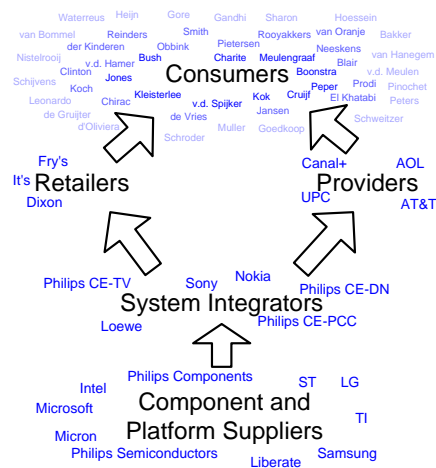


Figure 3.5: Example value chain

The consumer appliances itself are distributed through 2 different channels: the retailers and the service providers. Retailers sell appliances directly to the consumers, earning their money with this appliance sales and sometimes also with maintenance contracts for these appliances. Providers sell services (for instance telecom, internet), where the appliance is the means to access these services. The providers earn their money via the recurring revenues of the services.

Retailers and service providers have entirely different business models, which will be reflected by differences in the key drivers for both parties.

Reality is even much more complicated. For instance adding the *content* providers to the value chain adds an additional set of business models, with a lot of conflicting interests (especially Digital Rights Management, which is of high importance for the content providers, but is often highly conflicting with (legal) consumer interests).

3.4 Suppliers

The value chain must be described from the point of view of the customer. The customer sees your company as one of the (potential) suppliers. From the customer point of view products from many suppliers have to be integrated to create the total solution for his needs.

In terms of your own company this means that you have to make a map of

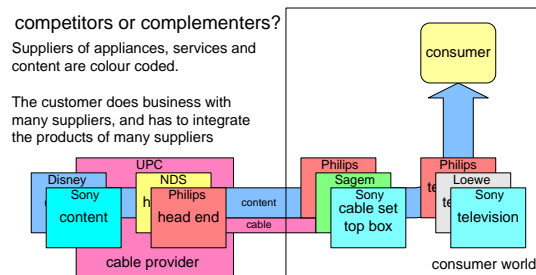
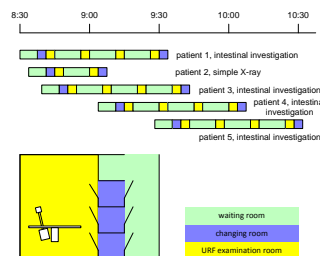


Figure 3.6: Example of simple supplier map for a cable provider

competitors and complementers, which together will supply the solution to the customer. Figure 3.6 shows an example of a simple supplier map for a cable provider. If your company is delivering set top boxes, then some companies can be viewed as competitor and complementer at the same time.

Chapter 4

The application view



4.1 Introduction

The application view is used to understand how the customer is achieving his objectives. The methods and models used in the application view should discuss the customer's world. Figure 4.1 shows an overview of the methods discussed here.

The customer is a gross generalization, which can be made more specific by identifying the customer stakeholders and their concerns, see section 4.2.

The customer is operating in a wider world, which he only partially controls. A context diagram shows the context of the customer, see section 4.3. Note that part of this context may interface actively with the product, while most of this context simply exists as neighboring entities. The fact that no interface exists is no reason not to take these entities into account, for instance to prevent unwanted duplication of functionality.

The customer domain can be modelled in static and dynamic models. Entity relationship models (section 4.4) show a static view on the domain, which can be complemented by dynamic models (section 4.5).

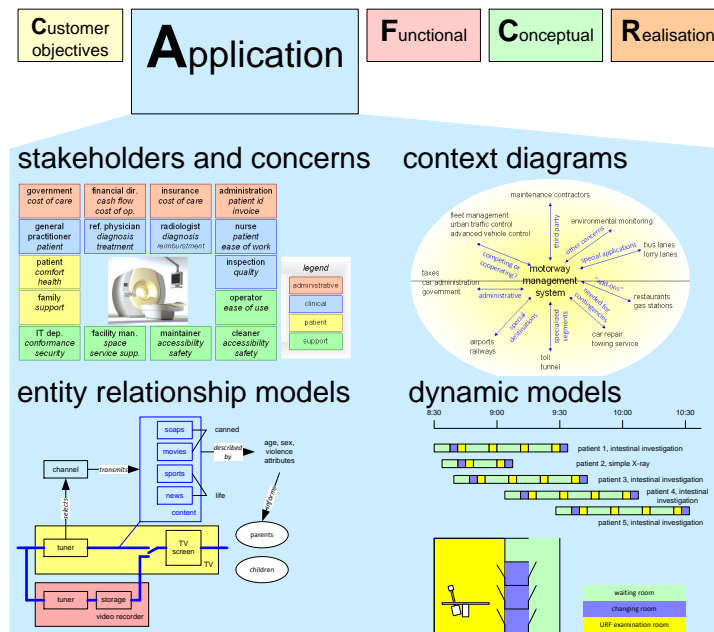


Figure 4.1: Overview of methods and models that can be used in the application view

4.2 Customer stakeholders and concerns

In the daily use of the system many human and organizational entities are involved, all of them with their own interests. Of course many of these stakeholders will also appear in the static entity relationship models. However human and organizations are very complex entities, with psychological, social and cultural characteristics, all of them influencing the way the customer is working. These stakeholders have multiple concerns, which determine their needs and behavior. Figure 4.2 shows stakeholders and concerns for an MRI scanner.

The IEEE 1471 standard about architectural descriptions uses stakeholders and concerns as the starting point for an architectural description.

Identification and articulation of the stakeholders and concerns is a first step in understanding the application domain. The next step can be to gain insight in the *informal* relationships. In many cases the formal relationships, such as organization charts and process descriptions are solely used for this view, which is a horrible mistake. Many organizations function thanks to the unwritten information flows of the social system. Insight in the informal side is required to prevent a solution which does only work in theory.

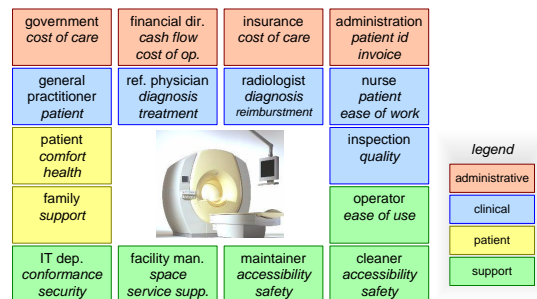


Figure 4.2: Stakeholders and concerns of an MRI scanner

4.3 Context diagram

The system is operating in the customer domain in the context of the customer. In the customer context many systems have some relationship with the system, quite often without having a direct interface.

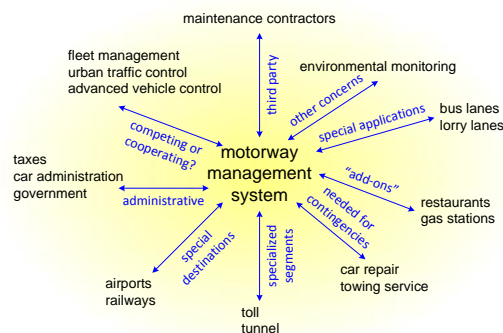


Figure 4.3: Systems in the context of a motorway management system

Figure 4.3 shows a simple context diagram of a motorway management system. Tunnels and toll stations often have their own local management systems, although they are part of the same motorway. The motorway is connecting destinations, such as urban areas. Urban areas have many traffic systems, such as traffic management (traffic lights) and parking systems. For every system in the context questions can be asked, such as:

- is there a need to interface directly (e.g. show parking information to people still on the highway)
- is duplication of functionality required (measuring traffic density and sending it to a central traffic control center)

4.4 Entity relationship model

The OO (Object Oriented software) world is quite used to entity relationship diagrams. These diagrams model the outside world in such a way that the system can interact with the outside world. These models belong in the "CAFCR" thinking in the conceptual view. The entity relationship models advocated here model the customers world in terms of entities in this world and relations between them. Additionally also the activities performed on the entities can be modelled. The main purpose of this modelling is to gain insight in how the customer is achieving his objectives.

One of the major problems of understanding the customers world is its infinite size and complexity. The art of making an useful entity relationship model is to very carefully select what to include in the model and therefore also what **not** to include. Models in the application view, especially this entity relationship model, are by definition far from complete.

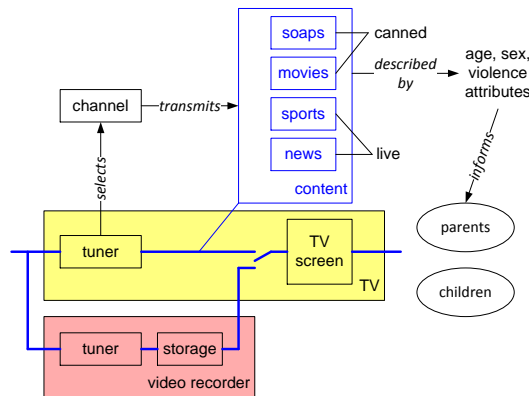


Figure 4.4: Diagram with entities and relationship for a simple TV appliance

Figure 4.4 shows an example of an entity relationship model for a simple TV. Part of the model shows the well recognizable flow of video content (the bottom part of the diagram), while the top part shows a few essential facts about the contents. The layout and semantics of the blocks are not strict, these form-factors are secondary to expressing the essence of the application.

4.5 Dynamic models

Many models, such as entity relationship models, make the static relationships explicit, but don't address the dynamics of the system. Many different models can be used to model the dynamics, or in other words to model the behavior in time. Examples of dynamic models are shown in figure 4.5

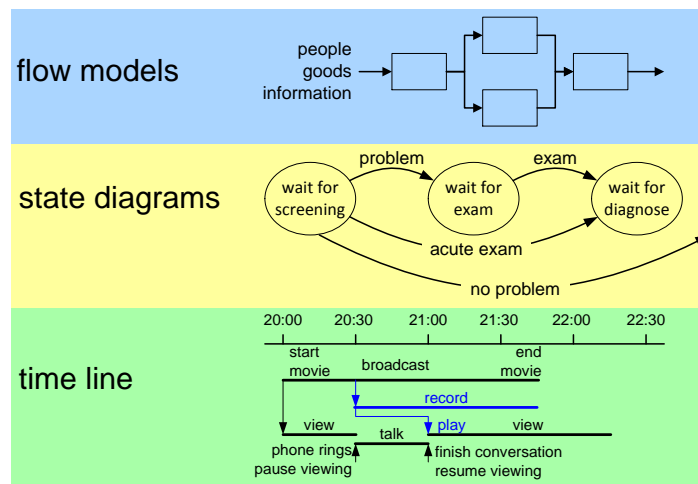


Figure 4.5: Examples of dynamic models

Productivity and Cost of ownership models are internally based on dynamic models, although the result is often a more simplified parameterized model, see figure 4.6.

Figure 4.7 shows an example of a time-line model for an URF examination room. The involved rooms play an important role in this model, therefore an example geographical layout is shown to explain the essence of the time-line model.

The patient must have been fasting for an intestine investigation. In the beginning of the examination the patient gets a barium meal, which slowly moves through the intestines. About every quarter of an hour a few X-ray images-images are made of the intestines filled with barium. This type of examination is interleaving multiple patients to efficiently use the expensive equipment and clinical personnel operating it.

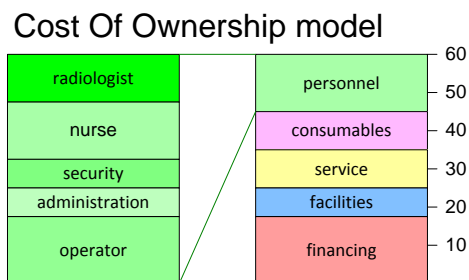
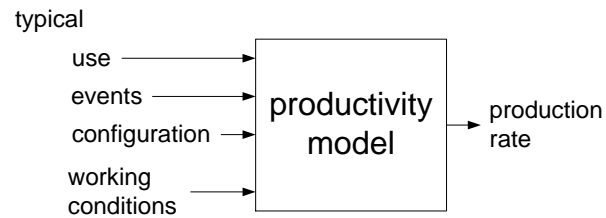


Figure 4.6: Productivity and cost models

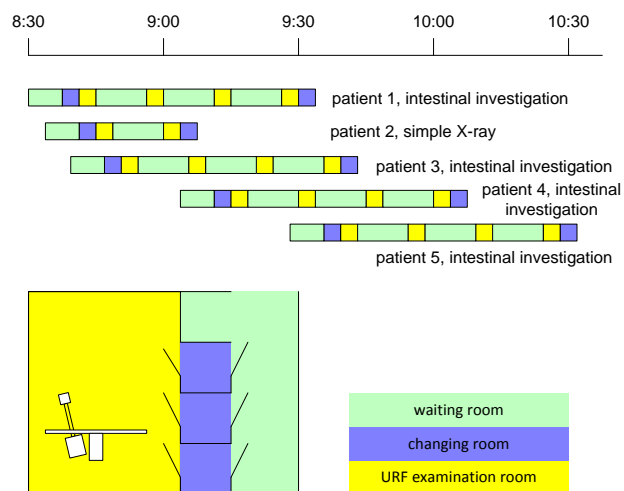
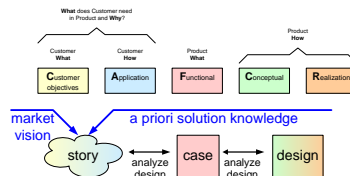


Figure 4.7: Dynamics of an URF examination room

Chapter 5

Story How To



5.1 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 5.1 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.

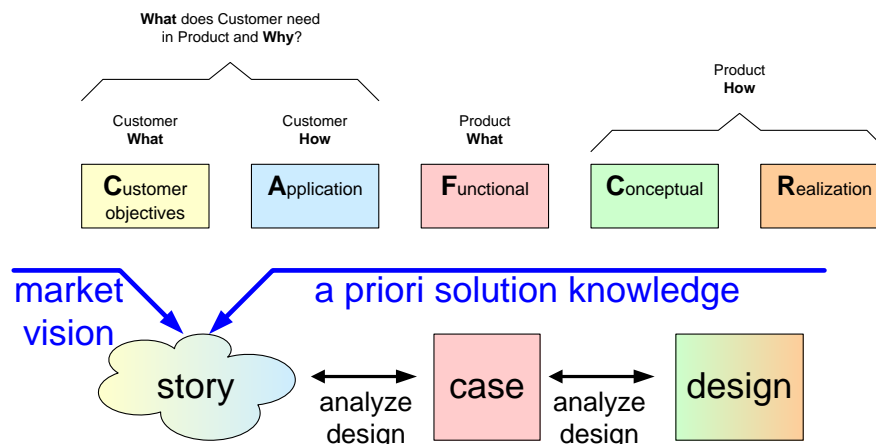


Figure 5.1: From story to design

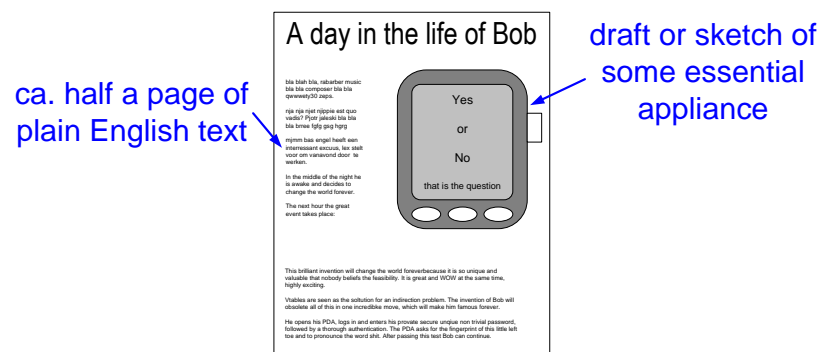


Figure 5.2: Example story layout

5.2 How to Create a Story?

A story is a short single page story, as shown in Figure 5.2, 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.

5.3 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.

5.4 Criteria

Figure 5.3 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.

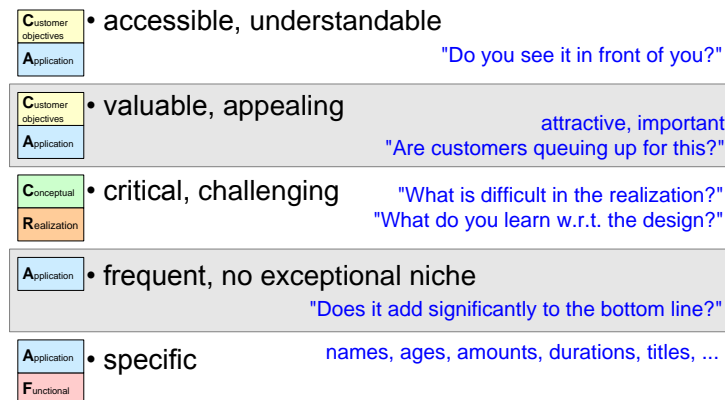


Figure 5.3: criteria for a good story

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.

5.5 Example Story

Figure 5.4 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 5.5 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 rumble 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 5.4: Example of a story

5.6 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.

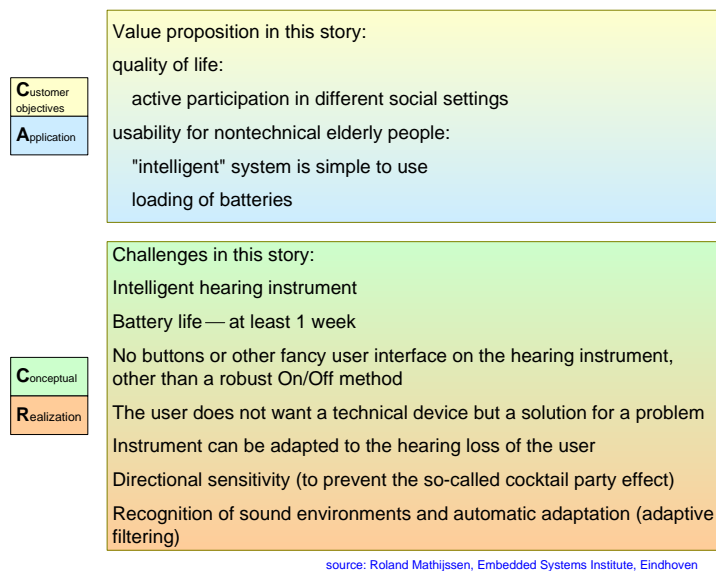
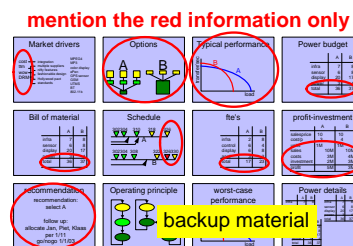


Figure 5.5: Value and Challenges in this story

Chapter 6

How to present architecture issues to higher management



6.1 Introduction

The architect bridges the technology world with the other business related worlds, by understanding these other worlds and by having ample know-how of technologies. Management teams are responsible for the overall business performance, which in the end is expressed in financial results.

Many architects and management teams are captured in a vicious circle:

- architects complain about management decisions and lack of know-how of managers
- managers complain about lack of input data and invisible architects

One way to break this vicious circle is to improve the managerial communication skills of architects. We address a frequently needed skill: presenting an architecture issue to a management team.

The architect should contribute to the managerial decision process by communicating technology options and consequences of technological decisions. Figure 6.1

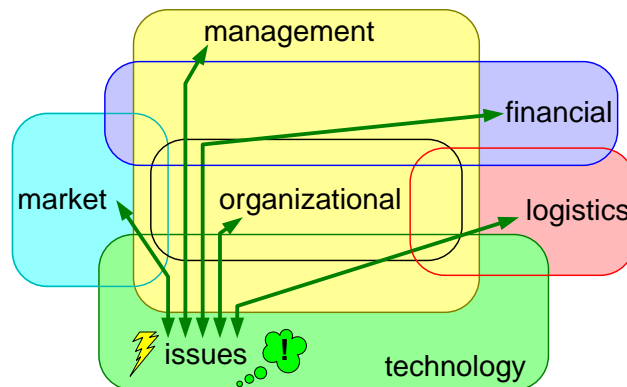


Figure 6.1: Architectural issues related to managerial viewpoints

shows a number of the relevant, somewhat overlapping, viewpoints. The figure indicates what links architects should communicate to management teams.

<i>common characteristics</i>	<i>highly variable characteristics</i>
+ action-oriented	? technology knowledge
+ solution rather than problem	from extensive to shallow
+ impatient, busy	? style from power play to
+ want facts not beliefs	inspirational leadership
+ operate in a political context	
+ bottom-line oriented: profit, return on investment, market share, etc.	

Figure 6.2: Characteristics of managers in higher management teams

Architects must have a good understanding of their target audience. Figure 6.2 characterizes the managers in management teams. Their main job is to run a healthy business, which explains many of these characterizations: *action oriented*, *solution rather than problem*, *impatient*, *busy*, *bottom-line oriented*: profit, return on investment, market share, et cetera, and *want facts not believes*. These managers operate with many people all with their own personal interests. This means that managers *operate in a political context* (something which architects like to ignore).

Some characteristics of management teams depend on the company culture. For example, the amount of technology know-how can vary from extensive to shallow. Or, for example, the management style can vary from power play to inspirational leadership.

6.2 Preparation

Presentations to higher management teams must always be prepared with multiple people: a small preparation team. The combined insights of the preparation team enlarge the coverage of important issues, both technical as well as business. the combined understanding of the target audience is also quite valuable. Figure 6.3 shows how to prepare the content of the presentation as well as how to prepare for the audience.

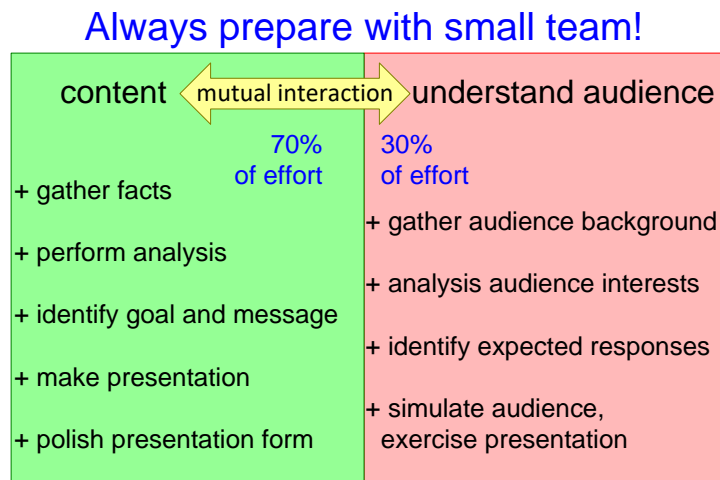


Figure 6.3: How to prepare

The content of the presentation must be clear, address the main issues, and convey the message, see also 6.3. The message must have substance for managers, which means that it should be *fact based*. The first steps are *gathering facts* and *performing analysis*. Based on these facts the *goal* and *message* of the presentation must be articulated. All this information must be combined in a *presentation*. When the presentation content is satisfactory the form must be polished (templates, colors, readability, et cetera). Although this has been described as a sequential process, the normal incremental spiral approach should be followed, going through these steps in 2-3 passes.

The members of management teams operate normally in a highly political context, mutually as well as with people in their context. This politics interferes significantly with the decision making. The political situation should be mapped by the preparation team, the political forces must be identified and understood. This is done by *analyzing the audience*, their *background* and their *interests*. The preparation team can gain a lot of insight by discussing the *expected responses* of the management team. At some moment the preparation team can *simulate* (role-play) the management team in a proof-run of the presentation. The understanding of the

audience must be used to select and structure the content part of the presentation. This activity should be time-multiplexed with the content preparation; 70% of the time working on content, 30% of the time for reflection and understanding of the audience.

6.3 The presentation material

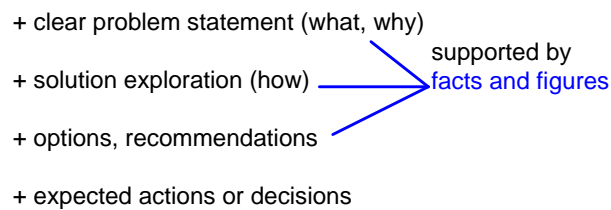


Figure 6.4: Recommended content

Figure 6.4 provides guidelines for the contents of the presentation. A clear *problem statement* and an *exploration of solution(s)* should address the technical issues as well as the translation to the business consequences. Normally a range of options are *provided*. The options are *compared* and *recommendations* are provided. Note that options that are unfavorable from architectural point of view are nevertheless options. It is the challenge for the architect to articulate why these options are bad and should not be chosen. Architect enable and streamline the decision making by providing clear recommendations and by indicating what *actions* or *decisions* are required.

All content of the presentation should be to the point, *factual* and *quantified*. Quantified does not mean certain, often quite the opposite, future numbers are estimates based on many assumptions. The reliability of the information should be evident in the presentation. Many facts can be derived from the past. *Figures* from the past are useful to “calibrate” future options. Deviations from trends in the past are suspect and should be explained.

The presentation material should cover more than is actually being presented during the presentation itself. Some supporting data should be present on the sheets, without mentioning the data explicitly during the presentation. This allows the audience to assess the validity of the presented numbers, without the need to zoom in on all the details.

It is also useful to have additional backup material available with more in depth supporting data. This can be used to answer questions or to focus the discussion: speculation can be prevented by providing actual data.

The use of demonstrators and the show of artifacts (components, mock-ups)

mention the red information only

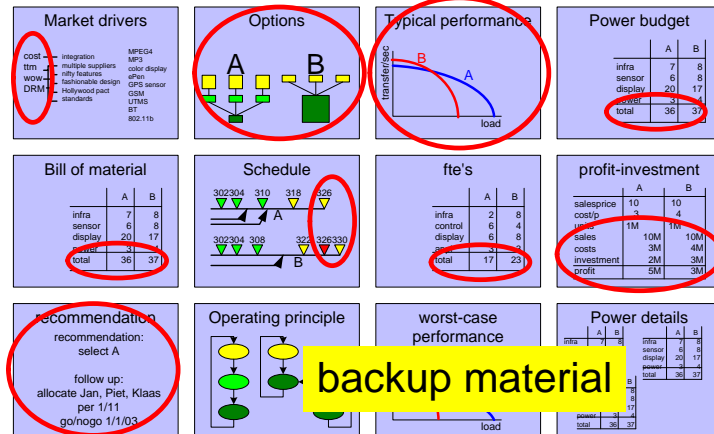


Figure 6.5: Mentioned info, shown info and backup info

makes the presentation more lively. The demonstrations should be short and attractive (from customer point of view), while illustrating the value and technological possibilities and issues.

poor form can easily distract from purpose and content

presentation material

+ professional

+ moderate use of
color and animations

+ readable

+ use demos and show artifacts

presenter's appearance

+ well dressed

+ self confident but open

but stay yourself,
stay authentic

Figure 6.6: Form is important

Architects prefer to focus on the content, form is supportive to transfer the content. However architects should be aware that managers can be distracted by the form of a presentation, potentially spoiling the entire meeting by small issues. Figure 6.6 gives a number of recommendations with respect to the form of the presentation and the appearance of the presenter.

The presentation material (slides, demonstrators, video, drawings, et cetera)

has to look professional. Slides will use color and other presentation features. However, moderation in the use of colors, animations and other presentation features is recommended; an overload of these colors and features does not look professional and will distract the audience from the actual content. Information on the slides has to be readable: use large enough fonts and use sufficient contrast with the background. Pay special attention to quality and readability, when copy-pasting information from other sources. Sometimes it is better to recreate a high quality table or graph than to save effort by copy-pasting an unreadable table or graph.

The appearance of the presenter can also make or break the presentation. The presenter should give sufficient attention to clothes and overall appearance. Don't exaggerate this, you should stay yourself and still be authentic. Other people immediately sense it when the appearance is too exaggerated, which is also damaging for your image.

6.4 The Presentation

<i>do not</i>	<i>do</i>
- preach beliefs	+ quantify, show figures and facts
- underestimate technology knowledge of managers	+ create faith in your knowledge
- tell them what they did wrong	+ focus on objectives
- oversell	+ manage expectations

Figure 6.7: Don't force your opinion, understand the audience

Figures 6.7 and 6.8 show in the *don't* column a number of pitfalls for an architect when presenting to higher management teams. The preferred interaction pattern is given in the *do* column.

The pitfalls in Figure 6.7, *preaching beliefs*, *underestimating know-how of managers*, and *telling managers what they did wrong*, are caused by insufficient understanding of the target audience. In these cases the opinion of the architect is too dominant, opinions work counterproductive. *Overselling* creates a problem for the future: expectations are created that can not be met. The consequence of overselling is loss of credibility and potentially lack of support in tougher times. Architects must *manage* the *expectations* of the audience.

When presenting the architect tries to achieve multiple objectives:

- Create awareness of the problem and potential solutions by *quantification* and by *showing figures and facts*.

- Show architecting competence in these areas, with the message being: “you, the manager, can delegate the technical responsibility to me”. This creates *faith* in the *architect’s know-how*.
- Facilitate decision making by translating the problem and solution(s) in business consequences, with the *focus on objectives*.

This means that sufficient technological content need to be shown, at least to create faith in the architect’s competence. Underestimation of the managerial know-how is arrogant, but mostly very dangerous. Some managers have a significant historic know-how, which enable them to assess strengths and weaknesses quickly. Providing sufficient depth to this type of manager is rewarding. The less informed manager does not need to fully understand the technical part, but at least should get the feeling that he or she understands the issues.

<i>do not</i>	<i>do</i>
- let one of the managers hijack the meeting	+ maintain the lead
- build up tensions by withholding facts or solutions	+ be to the point and direct
- be lost or panic at unexpected inputs or alternatives	+ acknowledge input, indicate consequences (facts based)

Figure 6.8: How to cope with managerial dominance

The impatience and action orientation of managers makes them very dominant, with the risk that they take over the meeting or presentation. Figure 6.8 shows a number of these risks and the possible counter measures:

Managers hijacking the meeting can be prevented by maintaining the lead as presenter.

Build up tensions by withholding facts or solutions, but be to the point and direct. For example, it can be wise to start with a summary of the main facts and conclusions, so that the audience know where the presentation is heading.

Be lost or panic at unexpected inputs or alternatives. Most managers are fast and have a broad perspective that helps them to come with unforeseen options. Acknowledge inputs and indicate the consequences of alternatives as far as you can see them (fact based!).

An example of an unexpected input might be to outsource a proposed development to a low-cost country. The outsourcing of developments of core components might

require lots of communication and traveling, creating costs. Such consequence has to be put on the table, but refrain from concluding that it is (im)possible.

6.5 Exercise

The SARCH course [7] on System Architecting contains an exercise, where the participants can apply then lessons learned by giving a presentation to a (simulated) management team. The presenter gives his presentation for the participants and the teacher, who play the role of this higher management team.

- + Bring a clear **architecture message** to
- + a **Management team** at least 2 hierarchical levels higher
- + with **10 minutes** for **presentation including discussion**
(no limitation on number of slides)
- * architecture message =
technology options in relation with **market/product**
- * address the **concerns** of the **management stakeholders**:
translation required from **technology** issues into
business consequences (months, fte's, turnover, profit, investments)

Figure 6.9: Exercise presentation to higher management

Figure 6.9 shows the description of this exercise. The group of participants is divided in 4 teams of about 4 people, preferably from the same domain. These teams have somewhat less than 2 hours for the preparation of the presentation. The exercise is explained to them several days before and the teams are also formed days before. This enables the team to determine a subject and message in a background process, during lunch and in the breaks. Determining the subject and message requires quite some elapsed time. It is highly recommended to take a subject from *real-life*: "What you always wanted to tell topmanagement".

Figure 6.10 shows the schedule of the exercise. Every presentation is 10 minutes sharp, **including** the interaction with the management team. Directly after the presentation feedback is given by the participants as well as by the teacher. This feedback should follow the normal feedback guidelines: mentioning the strong points, before discussing the options for improvement. The teacher must ensure that sufficient feedback is given, the material in this exercise can be used as guideline.

The limited preparation time implies that the result will also be limited. The form will be limited (handwritten flipovers) and most of the historical data will be made up.

The teacher should stimulate the complete group to really participate in the role

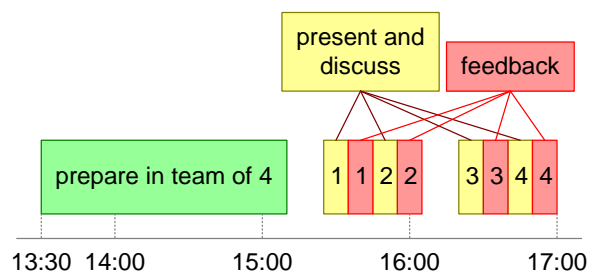
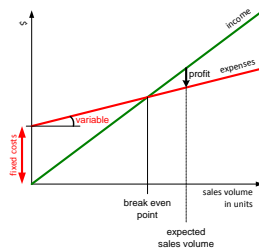


Figure 6.10: Schedule of the presentation exercise

play, it can also be a lot of fun.

Chapter 7

Simplistic Financial Computations for System Architects.



7.1 Introduction

Many system architects shy away from the financial considerations of the product creation. In this document a very much simplified set of models is offered to help the architect in exploring the financial aspects as well. This will help the architect to make a "sharper" design, by understanding earlier the financial aspects.

The architect should always be aware of the many simplifications in the models presented here. Interaction with real financial experts, such as controllers, will help to understand shortcomings of these models and the finesses of the highly virtualized financial world.

In Section 7.2 a very basic cost and margin model is described. Section 7.3 refines the model at the cost side and the income side. In Section 7.4 the time dimension is added to the model. Section 7.5 provides a number of criteria for making financial decisions.

7.2 Cost and Margin

The simplest financial model looks only at the selling price (what does the customer pay), the cost price (how much does the manufacturing of the product actually cost). The difference of the selling price and the cost price is the margin. Figure 7.1 shows these simple relations. The figure also adds some annotations, to make the notions more useful:

- the cost price can be further decomposed in material, labor and other costs
- the margin ("profit per product") must cover all other company expenses, such as research and development costs, before a real profit is generated
- most products are sold as one of the elements of a value chain. In this figure a retailer is added to show that the street price, as paid by the consumer, is different from the price paid by the retailer[1].

The annotation of the other costs, into transportation, insurance, and royalties per product, show that the model can be refined more and more. The model without such a refinement happens to be rather useful already.

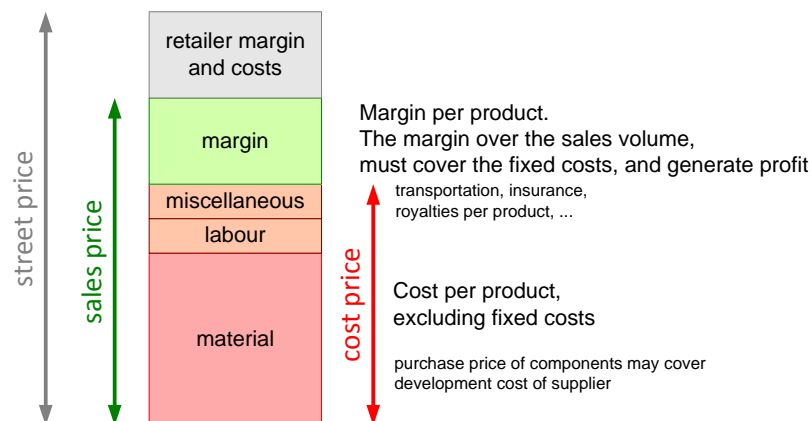


Figure 7.1: The relation between sales price, cost price and margin per product

The translation of margin into profit can be done by plotting income and expenses in one figure, as shown in Figure 7.2, as function of the sales volume. The slope of the expenses line is proportional with the costs per product. The slope of the income line is proportional with the sales price. The vertical offset of the expenses line are the fixed organizational costs, such as research, development, and overhead costs. The figure shows immediately that the sales volume must exceed the break even point to make a profit. The profit is the vertical distance between expenses

and income for a given sales volume. The figure is very useful to obtain insight in the robustness of the profit: variations in the sales volume are horizontal shifts in the figure. If the sales volume is far away from the break even point than the profit is not so sensitive for the the volume.

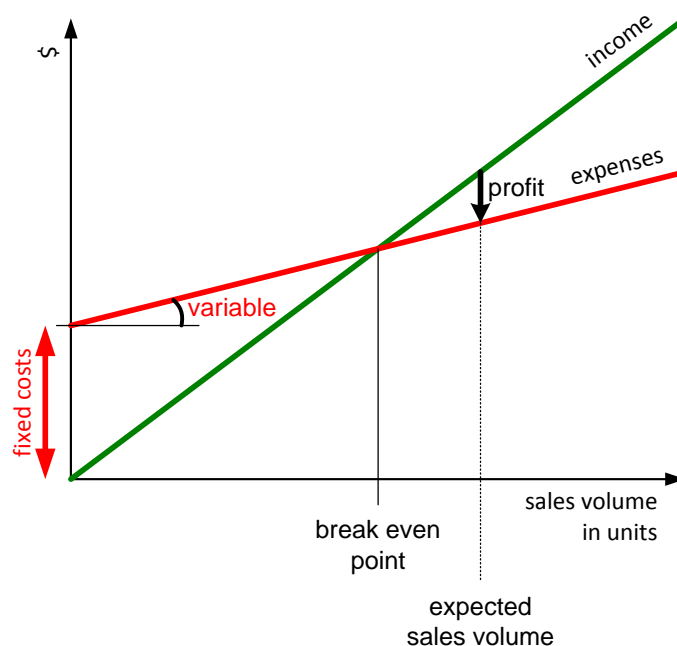


Figure 7.2: Profit as function of sales volume

7.3 Refining investments and income

The investments as mentioned before may be much more than the research and development costs only, depending strongly on the business domain. Figure 7.3 shows a decomposition of the investments. The R&D investments are often calculated in a simple way, by using a standard rate for development personnel that includes overhead costs such as housing, infrastructure, management and so on. The investment in R&D is then easily calculated as the product of the amount of effort in hours times the rate (=standardized cost per hour). The danger of this type of simplification is that overhead costs become invisible and are not managed explicitly anymore.

Not all development costs need to be financed as investments. For outsourced developments an explicit decision has to be made about the financing model:

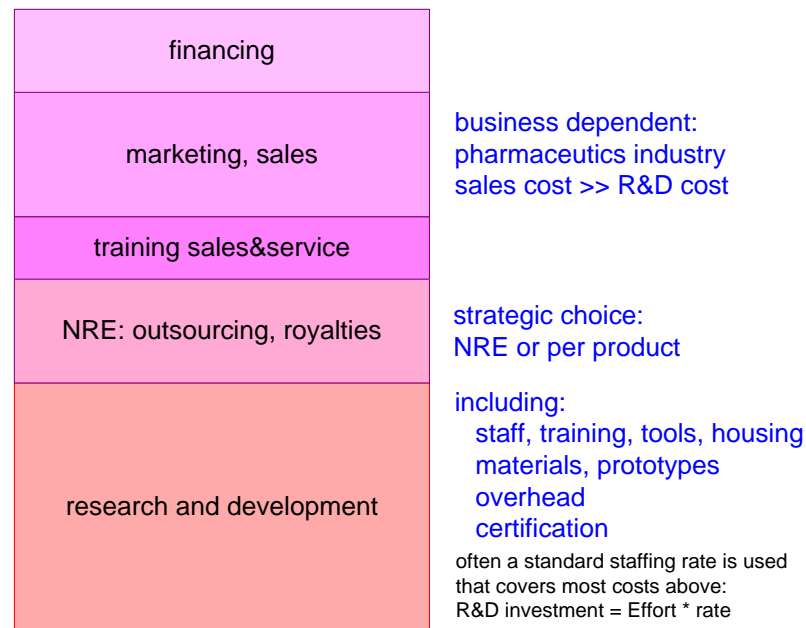


Figure 7.3: Investments, more than R&D

- the supplier takes a risk by making the investments, but also benefits from larger sales volumes
- the company pays the investment, the so called Non Recurring Engineering (NRE) costs. In this case the supplier takes less risks, but will also benefit less from larger sales volumes.

If the supplier does the investment than the development costs of the component are part of the purchasing price and become part of the material price. For the NRE case the component development costs are a straightforward investment.

Other investments to be made are needed to prepare the company to scale all customer oriented processes to the expected sales volume, ranging from manufacturing and customer support to sales staff. In some business segments the marketing costs of introducing new products is very significant. For example, the pharmaceutical industry spends 4 times as much money on marketing than on R&D. The financial costs of making investments, such as interest on the capital being used, must also be taken into account.

We have started by simplifying the income side to the sales price of the products. The model can be refined by taking other sources of income into account, as shown in Figure 7.4. The options and accessories are sold as separate entities, generating a significant revenue for many products. For many products the base products are

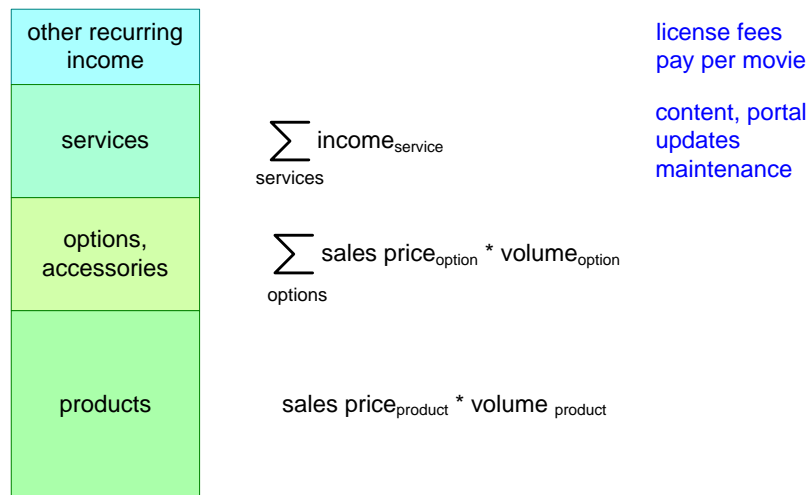


Figure 7.4: Income, more than product sales only

sold with a loss. This loss is later compensated by the profit on options and accessories.

Many companies strive for a business model where a recurring stream of revenues is created, for instance by providing services (access to updates or content), or by selling consumables (ink for prinjet printers, lamps for beamers, et cetera).

One step further is to tap the income of other players of the value chain. Example is the license income for MPEG4 usage by service and content providers. The chip or box supplier may generate additional income by partnering with the downstream value chain players.

7.4 Adding the time dimension

All financial parameters are a function of time: income, expenses, cash-flow, profit, et cetera. The financial future can be estimated over time, for example in table form as shown in Figure 7.5. This table shows the investments, sales volume, variable costs, income, and profit (loss) per quarter. At the bottom the accumulated profit is shown.

The cost price and sales price per unit are assumed to be constant in this example, respectively 20k\$ and 50k\$. The formulas for variable costs, income and profit are very simple:

$$\text{variable costs} = \text{sales volume} * \text{cost price}$$

$$\text{income} = \text{sales volume} * \text{sales price}$$

	Y1 Q1	Y1 Q2	Y1 Q3	Y1 Q4	Y2 Q1	Y2 Q2	Y2 Q3
investments	100k\$	400k\$	500k\$	100k\$	100k\$	60k\$	20k\$
sales volume (units)	-	-	2	10	20	30	30
material & labour costs	-	-	40k\$	200k\$	400k\$	600k\$	600k\$
income	-	-	100k\$	500k\$	1000k\$	1500k\$	1500k\$
quarter profit (loss)	(100k\$)	(400k\$)	(440k\$)	200k\$	500k\$	840k\$	880k\$
cumulative profit	(100k\$)	(500k\$)	(940k\$)	(740k\$)	(240k\$)	600k\$	1480k\$

cost price / unit = 20k\$
sales price / unit = 50k\$

*variable cost = sales volume * cost price / unit*
*income = sales volume * sales price / unit*
quarter profit = income - (investments + variable costs)

Figure 7.5: The Time Dimension

$$profit = income - (investments + variable costs)$$

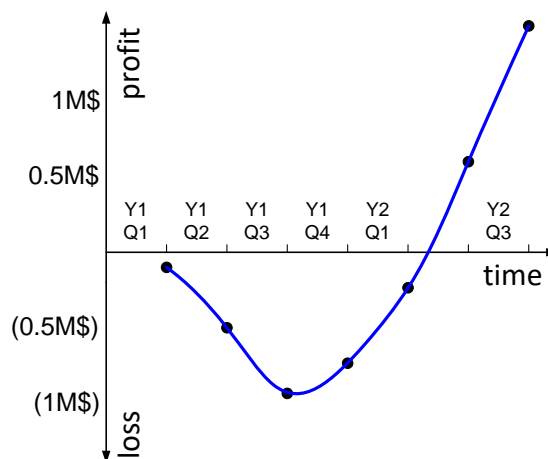


Figure 7.6: The “Hockey” Stick

Figure 7.6 shows the cumulative profit from Figure 7.5 as a graph. This graph is often called a “hockey” stick: it starts with going down, making a loss, but when the sales increase it goes up, and the company starts to make a profit. Relevant questions for such a graph are:

- when is profit expected?
- how much loss can be permitted in the beginning?
- what will the sustainable profit be in later phases?

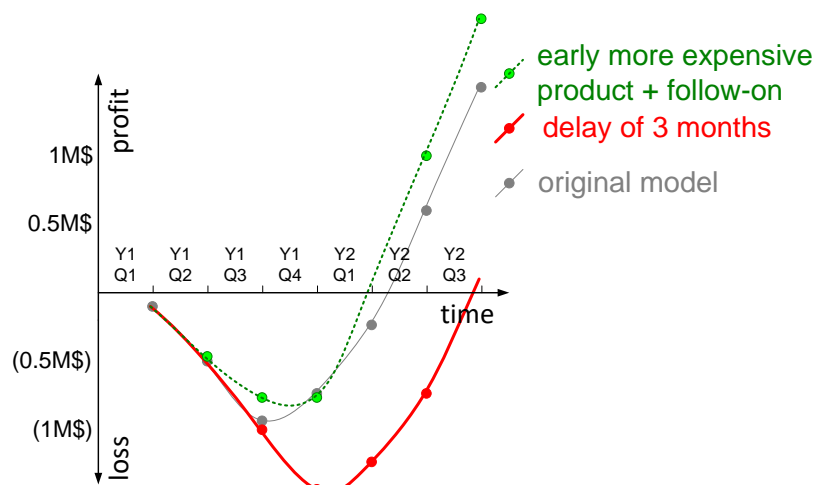


Figure 7.7: What if ...?

These questions can also be refined by performing a simple sensitivity analysis. Figure 7.7 shows an example of such an analysis. Two variations of the original plan are shown:

- a development delay of 3 months
- an intermediate more expensive product in the beginning, followed by a more cost optimized product later

The delay of 3 months in development causes a much later profitability. The investment level continues for a longer time, while the income is delayed. Unfortunately development delays occur quite often, so this delayed profitability is rather common. Reality is sometimes worse, due to loss of market share and sales price erosion. This example brings two messages:

- a go decision is based on the combination of the profit expectation and the risk assessment
- development delays are financially very bad

The scenario starting with a more expensive product is based on an initial product cost price of 30k\$. The 20k\$ cost price level is reached after 1 year. The benefit of an early product availability is that market share is build up. In this example the final market share in the first example is assumed to be 30 units, while in the latter scenario 35 units is used. The benefits of this scenario are mostly risk related. The loss in the beginning is somewhat less and the time to profit is somewhat better, but the most important gain is be in the market early and to reduce

the risk in that way. An important side effect of being early in the market is that early market feedback is obtained that will be used in the follow on products.

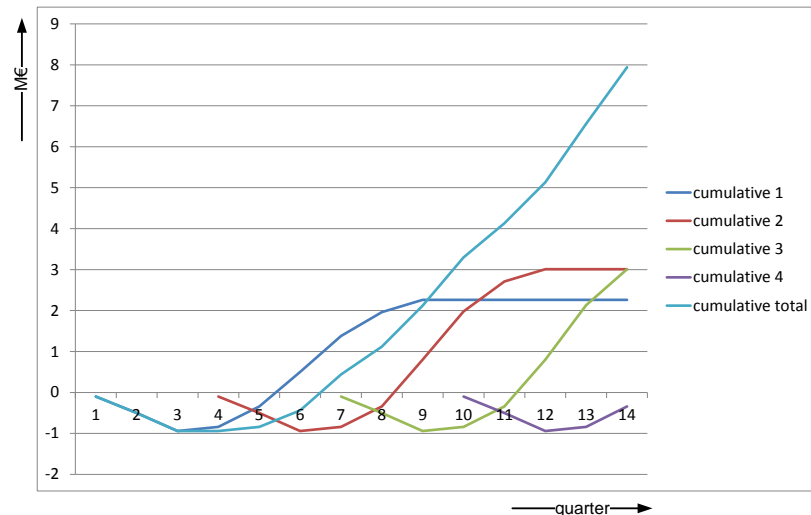


Figure 7.8: Stacking Multiple Developments

In reality, a company does not develop a single product or system. After developing an initial product, it will develop successors and may be expand into a product family. Figure reffig:SFCmultipleDevelopments shows how the cumulative profits are stacked, creating an integral hockey stick for the succession of products. In this graph the sales of the first product is reduced, while the sales of the second product is starting. This gradual ramp-up and down is repeated for the next products. The sales volume for the later products is increasing gradually.

7.5 Financial yardsticks

How to assess the outcome of the presented simple financial models? What are *good* scenarios from financial point of view? The expectation to be profitable is not sufficient to start a new product development. One of the problems in answering these questions is that the financial criteria appear to be rather dynamic themselves. A management fashion influences the emphasis in these criteria. Figure 7.9 shows a number of metrics that have been fashionable in the last decade.

The list is not complete, but it shows the many financial considerations that play a role in decision making.

Return On Investments is a metric from the point of view of the shareholder or the investor. The decision these stakeholders make is: what investment is the most attractive.

Return On Investments (ROI)

Net Present Value

Return On Net Assets (RONA) *leasing reduces assets, improves RONA*

turnover / fte *outsourcing reduces headcount, improves this ratio*

market ranking (share, growth) *"only numbers 1, 2 and 3 will be profitable"*

R&D investment / sales *in high tech segments 10% or more*

cash-flow *fast growing companies combine profits with negative cash-flow,
risk of bankruptcy*

Figure 7.9: Fashionable financial yardsticks

Return On Net Assets (RONA) is basically the same as ROI, but it looks at all the capital involved, not only the investments. It is a more integral metric than ROI.

turnover / fte is a metric that measures the efficiency of the human capital. Optimization of this metric results in a maximum added value per employee. It helps companies to focus on the core activities, by outsourcing the non-core activities.

market ranking (share, growth) has been used heavily by the former CEO of General Electric, Jack Welch. Only business units in rank 1, 2 or 3 were allowed. Too small business units were expanded aggressively if sufficient potential was available. Otherwise the business units were closed or sold. The growth figure is related to the shareholder value: only growing companies create more shareholder value.

R&D investment / sales is a metric at company macro level. For high-tech companies 10% is commonly used. Low investments carry the risk of insufficient product innovation. Higher investments may not be affordable.

cashflow is a metric of the actual liquid assets that are available. The profit of a company is defined by the growth of all assets of a company. In fast growing companies a lot of working capital can be unavailable in stocks or other non salable assets. Fast growing, profit making, companies can go bankrupt by a

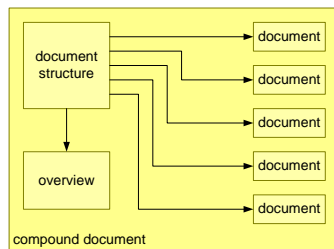
negative cash-flow. The crisis of Philips in 1992 was caused by this effect: years of profit combined with a negative cash-flow.

7.6 Acknowledgements

William van der Sterren provided feedback and references. Hans Barella, former CEO of Philips medical Systems, always stressed the importance of Figure 7.2, and especially the importance of a robust profit. Ad van den Langenberg pointed out a number of spelling errors.

Chapter 8

Granularity of Documentation



8.1 Introduction

Documentation is an important communication means in the Product Creation Process. The whole documentation set is written by multiple authors with different competencies. System architects contribute to the structure of the documentation, and write a small subset of the documentation themselves. The size of the units within the documentation structure is called the granularity of the documentation.

The right level of granularity improves the effectiveness of the documentation. We discuss criteria to design the documentation structure, the documentation granularity, and the documentation processes.

8.2 Stakeholders

Figure 8.1 shows the stakeholders of a document. The document is a description of some function or component that has to be realized by means of an implementation. The producers and the consumers of the function or component are the main stakeholders of the document. The author is also an important stakeholder. The function or component is always realized and used within a broader context. This context interacts with the function or component, so the persons responsible

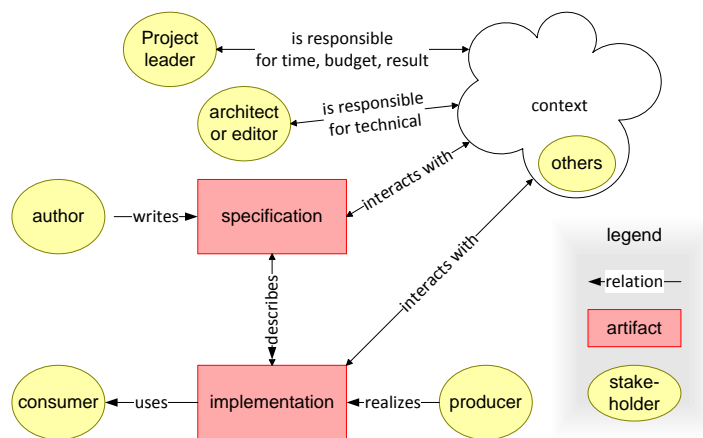


Figure 8.1: The stakeholders of a single document

for the context are also stakeholder of the document. In the context there will be other stakeholders as well; people who do have some involvement with the function or component.

8.2.1 Example digital flat screen TV

An electronics designer writes a specification for a Printed Circuit Board (PCB) to be used in a digital flat screen TV. A digital designer and a layout engineer realize the design, hence they are the producers. A software engineer will write the software making use of the functionality of the board, he is one of the consumers. The product (the digital flat screen TV) is the context for this PCB. The designer of the power supply might be a stakeholder, especially if the PCB has specific power requirements. The industrial designer responsible for the packaging is another stakeholder. The final product will have a project leader, responsible for the schedules, costs et cetera and is stakeholder with respect to these issues. The architect at last is responsible for a balanced and consistent product design, where the PCB should fit in.

8.3 Requirements

The documentation of a product need to be decomposed in smaller units, with the smallest units being atomic documents. We will discuss the requirements for the entire documentation structure, the documents itself, and the underlying process.

The criteria for the entire documentation structure and process are:

Accessibility for the readers ; the information should be understandable and readable

for the intended audience. The signal-to-noise ratio in the document must be high; information should not be hidden in a sea of words.

Low threshold for the readers ; No hurdles such as many pages of meta information, cumbersome security provisions, or complicated tools should dissuade readers from actually reading the document

Low threshold for the authors ; authors have to be encouraged to write. Hurdles, such as poor tools or cumbersome procedures, provide an excuse to delay writing.

Completeness of important information. Note that real completeness is an illusion, there are always more details that can be documented. All crucial aspects have to be covered by the entire documentation set.

Consistency of the information throughout the documentation. The writers strive for consistency, but we have to realize that in the complex world with many stakeholders some inconsistencies can be present. Inconsistencies that have significant impact on the result have to be removed.

Maintainability of the entire documentation, both during product creation as well as during the rest of the product life cycle.

Scalability of the documentation structure to later project phases, where many more engineers can be involved. The following measures help for scalability:

- well defined documentation structure
- explicit overview specifications at higher aggregation levels
- recursive application of structure and overview documents
- distribution of the review process

Evolvability of the documentation over time. Most documentation is re-used in successive projects.

Process to ensure the quality of the information . The quality of the content of the information is core to good results. Documentation that has been made only to satisfy the procedure is a waste of effort and time.

From reader point of view this translates in the requirements for the document infrastructure: it must be fast and easy to *view* and to *print* documents, and *searching* in the documentation also has to be fast and easy. Searching must be possible in a structured, e.g. hierarchical, way, and also via free text “a la Google”. Any part of the documentation must be reachable within a limited number of steps, so no excessively deep document hierarchies.

The criteria for the documents within the documentation structure are:

High cohesion within the document. The information in a document has to “belong” together. If information is not connected to the rest of the document, then this information might belong in another document.

Low coupling with other documents. Some coupling will be present, since the parts together will form the system. If the coupling is high, then the document decomposition is suspect and might need improvement.

Accessibility for the readers, as for the entire documentation.

Low threshold for the reader, as for the entire documentation.

Low threshold for the author, as for the entire documentation.

Manageable steps to create, review, and change the document. Documents in product creation are reviewed and updated frequently. Hence these operations should take limited effort and time. The consequence is that single documents should not be large.

Clear responsibilities, especially for the content of the document. Documents with multiple authors are suspect, responsibility for the content can be diffuse. Worse are documents where an anonymous team or committee is “the author”. If a document needs multiple authors, then it is often a symptom of bad decomposition. Also the reviewers responsibility must be clear, hence we recommend to limit the number of reviewers. When many reviewers are needed, then the decomposition is again suspect.

Clear position and relation with the context documents only make sense in the intended context. On purpose the information is captured in multiple documents. Therefore for every individual document it should be clear in what context it belongs and how it relates to other documents.

Well-defined status of the information. Documents are used and most valuable in the period when they are created. The content can be quite preliminary or draft. The document must clearly indicate what the status is of its content, so that readers can use it with proper precautions.

Timely availability of the document. When documents are too late available we do not harvest the value. Authors have to balance quality, completeness, and consistency against the required effort and time.

A very important function of documentation is communication. Communication requires that the information is accessible for all stakeholders, and that the threshold to produce documentation or to use documentation should be low¹.

¹Quite often organizations focus on the documentation procedures, and documentation

8.4 Documentation Structure

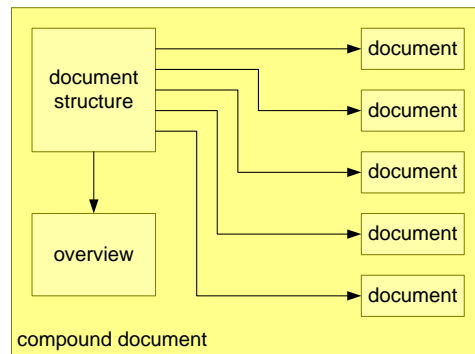


Figure 8.2: Large documents are decomposed in smaller documents, supported by a document structure and overview

The standard way to cope with large amounts of information is to decompose the information in smaller parts. The decomposition of the large amount of information results in a set of smaller documents. The structure of such a decomposition is made explicit in the “documentation structure”, fulfilling the requirement to have a *well defined documentation structure*. The documentation structure is managed as a normal document. An overview document is required to keep the overview accessible, addressing the requirement to have *overview specifications at higher aggregation levels*. Overviews help the readers, especially when the more detailed information gets scattered in smaller documents.

This decomposition is applied recursively, see Figure 8.3. In this way the granularity supports the realization of the requirements as described in the 8.3. For instance, the principle of *recursion* is a good answer to the requirements related to *scalability* of the entire documentation. Creating explicit structure and overview documents and allocating creation and maintenance to authors supports *maintainability*.

A fine grain structure, e.g. small documents, lower the threshold to make documents and to read the contents, in this way answering document requirements *accessibility for the reader, low threshold for the reader and low threshold for the author*.

The clarity and the value of the content is the foremost requirement for documentation. Decomposing the documentation is a balancing act in many dimensions, similar to the decomposition of systems. Clarity and value of the content may not

management, forgetting the main drivers mentioned in this subsection. The result can be tremendous thresholds, causing either apathy or bypasses. It cannot be stressed enough that procedures and tools are the **means** to solve a problem and not a goal in itself

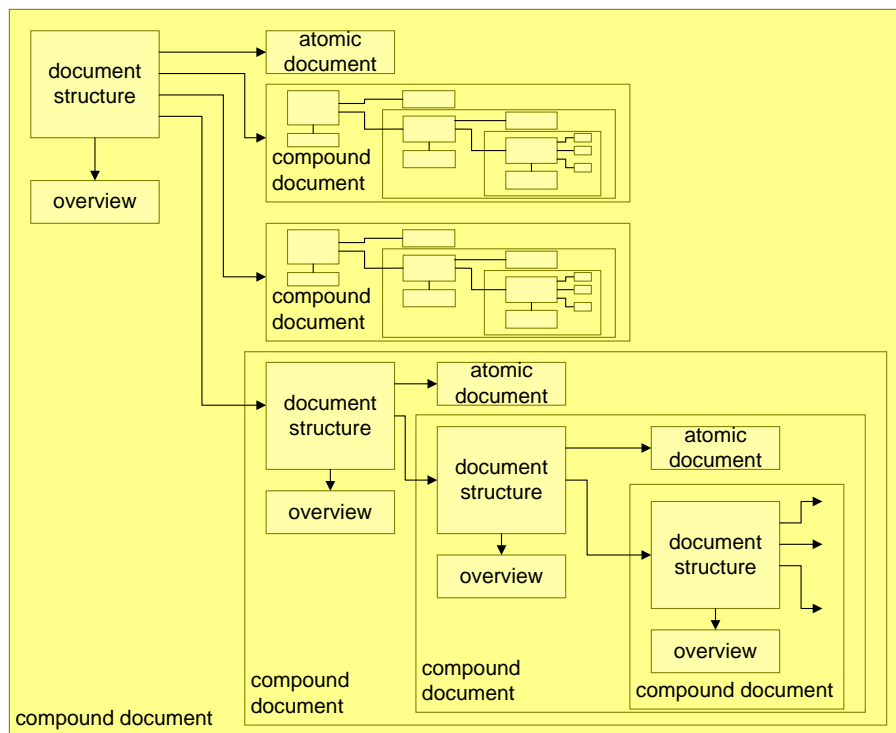


Figure 8.3: Decomposition is applied recursively until the atomic documents fulfill the requirements in section 8.3

suffer from the structure. Dogmatic structuring rules might be conflicting with clear responsibilities (*single author*). When authors write outside their expertise area, then there is a severe quality risk. The decomposition has to result in sufficiently small documents to support the requirement *Manageable steps to create, review, and change*. Large, monolithic documents violate this requirement.

The document granularity is an important design criterion for the documentation structure. The extreme that every *single value* is an entity² is not optimal, because the relations between values are even more important than the value itself. In case of *single value* documentation, relations are lost. The other extreme, to put everything in a single document, is conflicting with many of the requirements, such as *manageability*, *clear responsibilities*, *well-defined status* and *timely availability*. The granularity aspect, with the many psychological factors involved, is further discussed in 8.5.

² A common pitfall is to store all values in a database. In this way every value is an entity in itself. Such a database creates the suggestion of completeness and flexibility, but in reality it becomes a big heap, where the designers lose the overview. These databases may help the verification process, but do not fulfill the documentation needs.

8.5 Payload, the ratio between overhead and content

An atomic document must be small enough to be accessible to readers. Thick documents are put on top of the stack of “interesting papers to be read”, to be removed when this stack overflows. For most people time is the most scarce resource. Struggling through all kinds of overhead is a waste of their scarce and valuable time. Documentation effectively supports communication if the reader can start directly with reading the relevant information. Figure 8.4 shows the layout of a good document.

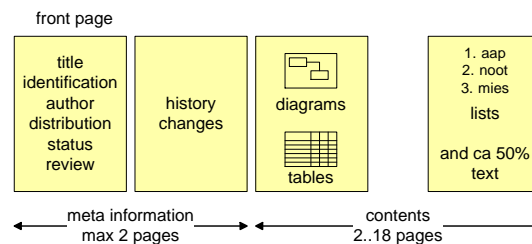


Figure 8.4: Layout of a good document, heuristic for the number of pages of a good document is $4 \leq nrofpages \leq 20$

The front page is used for all relevant meta-information. Meta-information is the information required for the document management, defining the status, responsibilities, context etc. The history and change information on the second page should be a service to the readers, to enable them to quickly see the relevant changes relative to earlier versions they might have read. More extensive change information, required for quality assurance purposes can be present in the document management system, it should not distract the reader from the information itself.

Such a document needs only to be opened to access the contents. Many older organizations tend to make documents with up to 10 pages of overhead information. Many people are interrupted by phone, calendar, e-mail, or person before reaching page three. The overhead de facto inhibits people to read the contents of badly written documents³.

The contents of a well written document ought to be optimized to get the essential information transferred. The reader community exists of different people, with differing reading and learning styles. To get information across the information must be visualized (diagrams), structured and summarized (tables and lists) and, to a limited extend, explained in text.

Once a document start its life cycle, the next risk is that the document keeps growing Authors have the tendency to transform comments and critiques of readers

³Often the situation is much worse than described here. In name of “standardization” these counterproductive layouts are made mandatory, forcing everyone to create thresholds for readers!

in explaining text. Unfortunately, large sections of text hide the key information, and violation of the maximum of 20 pages gets probable. It is better to translate the comments and critiques back into an improved diagram, table or list. Authors have to find the root cause of reader comments. For example an unclear diagram gives rise to misunderstanding.

Another frequent occurring trap is the extension of a document with missing context information. For instance, if the higher level specification is missing, parts of that specification are included in the lower level specification. An effective counter measure for this trap is to write the specification structure, showing the context and enabling to write the context later step by step. This strategy results in documents that are more focused, have a better cohesion internally, and have less coupling with other documents.

The heuristic mentioned in Figure 8.4 is that a good document should have 4 or more pages. This minimum should trigger people with the question if the information in a very small document has a right of existence on its own. The ratio overhead versus payload for very small documents is unbalanced. There are a small documents where the small size is appropriate.

The maximum number of pages for a good document is 20. These documents don't scare people away yet. A 20 page document can be read in less than one hour, and the review can also be done in less than one hour. For many purposes 10 to 15 page documents are optimal. If documents require more than 20 pages the recipe is simple: make it a compound document, so split the content in multiple smaller documents.

In large documents a natural split up is often directly visible.

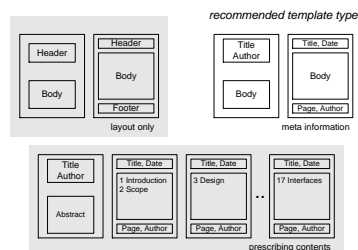
Large documents often violate a number of the requirements in 8.3. For instance, the document is edited by a single person but written by multiple authors. Another symptom of requirement violation is a document that is partly finished and partly in draft status (for instance "requirements" sections are written, while the "design" is still in full motion).

8.6 Acknowledgements

Angelo Hulshout triggered me to fill the the open ends in the requirements section.

Chapter 9

Template How To



9.1 Introduction

The introduction of a new process (way of working) is quite often implemented by supplying ready-to-go tools and templates. This implementation serves mainly the purpose of a smooth introduction of the new process.

Unfortunately the benefits of templates are often canceled by unforeseen side-effects, such as unintended application, inflexibility and so on. This intermezzo gives hints to avoid the **Template Trap**, so that templates can be used more effectively to support introduction of new processes.

Templates are used for all information based entities, such as documents, mechanical CAD designs and SW code. The information in this document applies to all these categories, although the text focuses on document templates.

9.2 Why Templates?

The rationale behind the use of a template is:

- Low threshold to apply a (new) process (1)
- Low effort to apply a (new) process (2)
- No need to know low level implementation details (3)
- Means to consolidate and reuse experiences (4)

Some common false arguments are:

- Obtain a uniform look (5)
- Force the application of a (new) process (6)
- Control the way a new process is applied (7)

Argument 5 is a bogus argument, uniformity¹ is not something to strive for, see section 9.9. Arguments 6 and 7 are the poor man's solution for lack of leadership and signals a dangerous disrespect for the target group.

9.3 New Process Introduction

Process Improvement drives result in enforcing existing processes or introduction of new processes. Any change introduces reactionary behavior ($action = -reaction$), urging the process improvement people to introduce the change in such a way that this reactionary behavior gives a minimal damage, see figure 9.1.

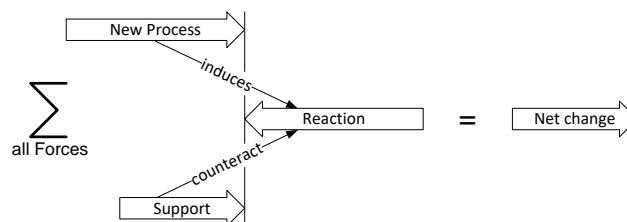


Figure 9.1: The reactionary force induced by the proposed new process is countered by giving support

The most frequent way to introduce a new process is to supply the means for the implementation of the process, in other words the emphasis is on the **how**, not

¹This will be elaborated in a future Intermezzo, *The Uniformity Trap*.

on the **why** nor on the **what**. Figure 9.2 shows the relation between a process and a template. The process itself focuses on **why** and **what** (and who and when), while the procedure, the tools and the template are the **how**.

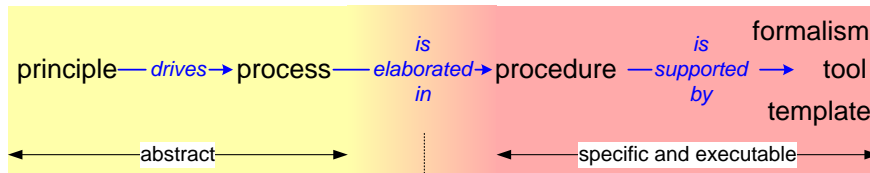


Figure 9.2: The relation between a template and a process

9.4 What does a Template contain

A template can support from *layout only* up to *complete contents standard*. Figure 9.3 shows a number of examples, Table 9.1 summarizes the characteristics. A layout only template does not have any notion of the information which will be in the document, nor does it presume anything about the process in which it is applied.

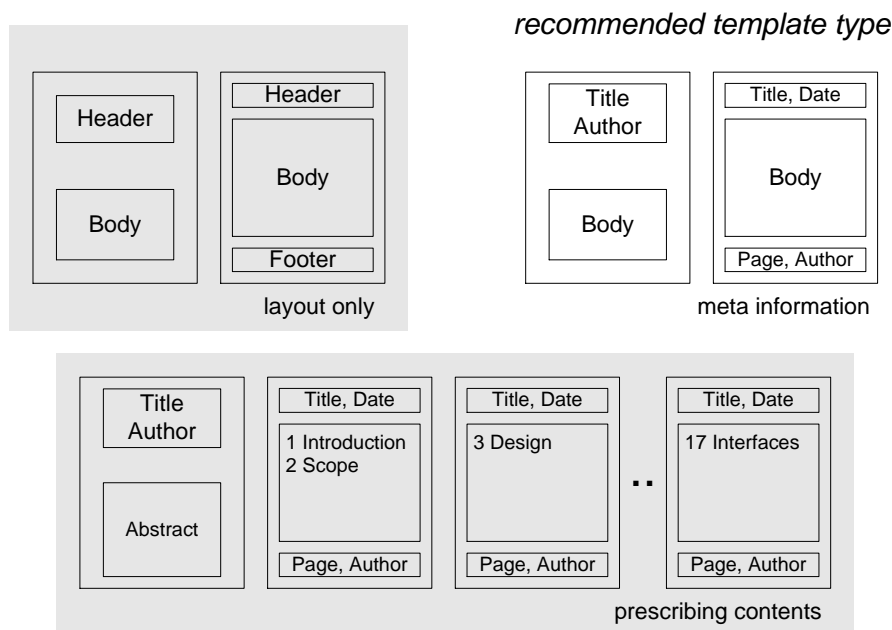


Figure 9.3: Templates from layout only, up to prescribing structure or contents, templates to support meta information are recommended.

A template with meta information supports the process in which it will be applied. The template has knowledge of the meta-information of the document and supports both the layout of the document as well as the presentation of the meta-information in this layout.

template type	context knowhow	value
layout only	no	low
meta information	process	high
prescribing content	process and domain	constraining

Table 9.1: Overview of Template characteristics

A template which prescribes the structure of the contents of the document has knowledge of the domain as well.

Recommendation: Use a template for layout and a minimum meta information set.

Avoid using a template to structure the contents. A documentation structure needs to be **designed**, see [8]. To help people in this design process of documentation guidelines containing checklists are effective. Templates invite people to generate "noisy" chapters (which should not have been present at all), or to write monolithic documents (because the entire checklist is present in one template). Guidelines with checklists at the other hand only mention contents, without suggesting any modularity yet.

Recommendation: Use checklists for structure and contents.

9.5 Copy Paste Modify Pattern

The understanding of the *copy paste modify pattern* will help to use templates effectively. The dominant implementation² strategy is the *copy paste modify pattern*:

- Look for a similar problem
- Copy its implementation
- Modify the copy to fulfil the new requirements

Majority of the work is to select the parts to be copied (or remove the unneeded parts) and to substitute the problem specific names, variables, functions et cetera.

A template is an optimization of this pattern in case of frequently reused implementations. The selection is performed once and the substitution is prepared to be easy.

²This holds for all information based implementations, from mechanical CAD drawings to management spreadsheets

9.6 Template Development

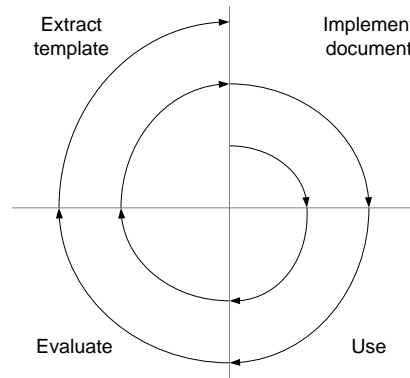


Figure 9.4: Spiral development model for Templates

The development of templates is basically the consolidation of experience. Once more a spiral development model best fits on the capturing of learning experiences, see figure 9.4. The motto is:

Use before Re-use

So implement a limited amount of documents (ca. 3), use these documents with other people, evaluate explicitly and extract then the template from these first documents. Implement the next set of documents on the basis of this template and repeat the same use, evaluation and extraction process. Keep repeating this forever!

9.7 Guidelines

A template is applied to support one or more processes. The deployment of the template is enabled by guidelines describing the way it should be used. The guidelines must be classified in mandatory rules and recommended practice.

An example of guidelines for meta information of a document is:

Mandatory on every page

- Author
- Title
- Status
- Version
- Date of last update
- Unique Identification
- Business Unit
- Page number

Mandatory on every document on top of the mandatory information per page

- Distribution (Notification) list
- Reviewers and commentators
- Document scope (Product family, Product, Subsystem, Module as far as applicable)
- Change history

Recommended Practice

- Short statement on frontpage stating what is expected from the addressed recipients, for example:
 - Please send comments before february 29, this document will be reviewed on that date
 - This document is authorized, changes are only applied via a change request
- See Granularity of Documentation [8] for guidelines for modularization and contents

The example defines a minimum mandatory set. No layout guideline is given except the fact that a subset of the meta information is mandatory on every page.

This illustrates a very important aspect of templates:

The procedure is mandatory, the template is only an enabling means, which means that anyone can make its own template as long as it fulfills the mandatory rules of the procedure.

9.8 Pitfalls

The most frequent pitfalls in the application of templates are:

- Author follows template instead of considering the purpose of the document.
- Template is too complex.
- There is an unmanageable number of variants.
- Mandatory use of templates results in:
 - no innovation of templates (= no learning)
 - no common sense in deployment
 - strong dependency on templates

A tendency exists to put a lot of information and intelligence in a template. For instance specialized Word templates, which prompt for the required fields. These kinds of templates are very vulnerable with respect to tools and environment. Changes in tools, environment or process play havoc with these nice looking templates. Good templates are, as good designs, simple. Simple templates are easily understood and easily modified, providing flexibility, room for innovation and room for common sense by customization to the problem.

In due time the amount of specialized templates grows. As in a normal design re-factoring is required to keep the overall set simple and consistent and hence maintainable.

The most common pitfall is to make the template mandatory instead of making the procedure mandatory. In other words the **how** is enforced instead of the **what**. This is the main cause of all the following pitfalls, such as no innovation, no common sense and a strong dependency on templates. The mandatory use of templates inhibits the innovation and common sense by individual users.

Recommendation: Enforce the procedure (*what*), provide the template (*how*) as supporting means.

9.9 Why I hate templates

Personally I hate templates. My way of working is based on immediate visual recognition of objects such as documents. For instance searching for a document on a large chaotic desktop is based on the visual image in my memory which is compared to the visual look of the documents on the desktop. When receiving a document the visual look immediately classifies the document with respect to author, project and status.

The uniformity caused by templates dramatically degrades my recognition performance and worse false matches turn up frequently.

This problem can be countered by allowing "personification" of documents, for instance by adding personal icons, images fonts et cetera. So by making variation on purpose!

Several people have pointed out to me that I violate my own needs for visual recognition with all Gaudí articles. Obviously here is room for improvement!

9.10 Summary

- Templates support (new) processes
- Use templates for layout and meta information support
- Do not use templates for documents structure or contents
- Stimulate evolution of templates, keep them alive
- Keep templates simple
- Standardize on **what** (process or procedure), not on **how** (tool and template)
- Provide (mandatory) guidelines and recommended practices
- Provide templates as a supportive choice, don't force people to use templates

9.11 Acknowledgements

Jürgen Müller identified the weak spots as usual, enabling an improved and more clear intermezzo. Discussion with Saar Muller helped to improve the terminology, such as guidelines and rules. The sharp eyes of Jaap v.d. Heijden helped to improve the figures.

Bibliography

- [1] Mark Abraham. Define and price for your market starting at end market values! <http://www.sticky-marketing.net/articles/pricing-for-channels.htm>, 2001.
- [2] Frederick P. Brooks. *The Mythical Man-Month*. Addison Wesley, 1975, ca. 1995.
- [3] Jean-Marc DeBaud and Klaus Schmid. A systematic approach to derive the scope of software product lines. In *21st international Conference on Software Engineering: Preparing for the Software Century*, pages 34–47. ICSE, 1999.
- [4] J. C. DeFoe (Editor). An identification of pragmatic principles. <http://www.incose.org/workgrps/practice/pragprin.html>, 1999.
- [5] INCOSE. International council on systems engineering. <http://www.incose.org/toc.html>, 1999. INCOSE publishes many interesting articles about systems engineering.
- [6] James N. Martin. *Systems Engineering Guidebook*. CRC Press, Boca Raton, Florida, 1996.
- [7] Gerrit Muller. CTT course SARCH. <http://www.gaudisite.nl/SARCHcoursePaper.pdf>, 1999.
- [8] Gerrit Muller. Granularity of documentation. <http://www.gaudisite.nl/DocumentationGranularityPaper.pdf>, 1999.
- [9] Gerrit Muller. Positioning the system architecture process. <http://www.gaudisite.nl/PositioningSystemArchitectureProcessPaper.pdf>, 1999.
- [10] Gerrit Muller. Requirements capturing by the system architect. <http://www.gaudisite.nl/RequirementsPaper.pdf>, 1999.
- [11] Gerrit Muller. Roadmapping. <http://www.gaudisite.nl/RoadmappingPaper.pdf>, 1999.

- [12] Gerrit Muller. The system architecture homepage. <http://www.gaudisite.nl/index.html>, 1999.
- [13] Eberhardt Rechtin and Mark W. Maier. *The Art of Systems Architecting*. CRC Press, Boca Raton, Florida, 1997.

History

Version: 0.1, date: May 28, 2004 changed by: Gerrit Muller

- Created very preliminary bookstructure from available Gaudí articles, no changelog yet

Version: 0, date: September 19, 2003 changed by: Gerrit Muller

- Created very preliminary bookstructure from available Gaudí articles, no changelog yet