

Module Requirements



Gerrit Muller
University of South-Eastern Norway-NISE
Hasbergsvei 36 P.O. Box 235, NO-3603 Kongsberg Norway
gaudisite@gmail.com

Abstract

This module addresses requirements: What are requirements? How to find, select, and consolidate requirements?

Distribution

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

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

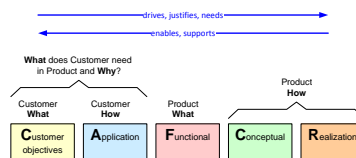
Contents

1	Short introduction to basic “CAFRCR” model	1
1.1	Introduction	1
1.2	The CAFRCR model	2
1.3	Who is the customer?	2
1.4	Life Cycle view	3
2	Fundamentals of Requirements Engineering	6
2.1	Introduction	6
2.2	Definition of Requirements	6
2.3	System as a black box	8
2.4	Stakeholders	8
2.5	Requirements for Requirements	9
3	Key Drivers How To	11
3.1	Introduction	11
3.2	Example Motor Way Management	11
3.3	CAF-views and Key Drivers	13
4	Requirements Elicitation and Selection	17
4.1	Introduction	17
4.2	Viewpoints on Needs	17
4.3	Requirements Value and Selection	19
5	The customer objectives view	22
5.1	Introduction	22
5.2	Key drivers	23
5.3	Value chain and business models	25
5.4	Suppliers	26
6	The application view	28
6.1	Introduction	28
6.2	Customer stakeholders and concerns	29

6.3 Context diagram 30
6.4 Entity relationship model 31
6.5 Dynamic models 31

Chapter 1

Short introduction to basic “CAFRCR” model



1.1 Introduction

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

An early tutorial[3] of this model used the concatenation of the first letters of the views in this model to form the acronym “CAFRCR” (Customer Objectives, Application, Functional, Conceptual, Realization). This acronym is used so often within the company, that changing the acronym has become impossible. We keep the name constant, despite the fact that better names for some of the views have been proposed. The weakest name of the views is *Functional*, because this view also contains the so-called *non functional* requirements. A better name for this view is the Black-Box view, expressing the fact that the system is described from outside, without assumptions about the internals.

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

1.2 The CAFCR model

A useful top level decomposition of an architecture is provided by the so-called “CAFCR” model, as shown in figure 1.1. The *Customer Objectives* view and the *Application* view provide the **why** from the customer. The *Functional* view describes the **what** of the product, which includes (despite the name) also the *non-functional* requirements. The **how** of the product is described in the *Conceptual* and *Realization* view, where the conceptual view is changing less in time than the fast changing realization (Moore’s law!).

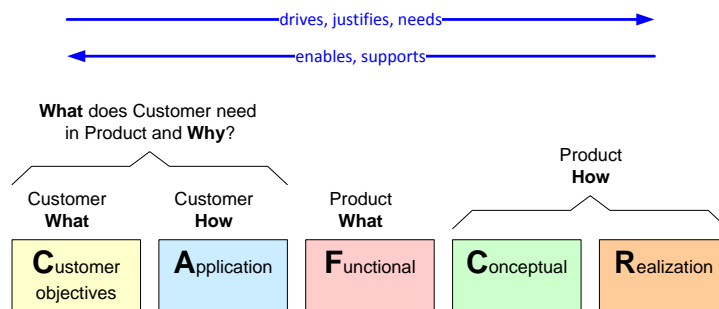


Figure 1.1: The “CAFCR” model

The job of the architect is to integrate these views in a consistent and balanced way. Architects do this job by *frequent viewpoint hopping*: looking at the problem from many different viewpoints, sampling the problem and solution space in order to build up an understanding of the business. Top-down (objective driven, based on intention and context understanding) in combination with bottom-up (constraint aware, identifying opportunities, know how based), see figure 1.2.

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

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

1.3 Who is the customer?

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

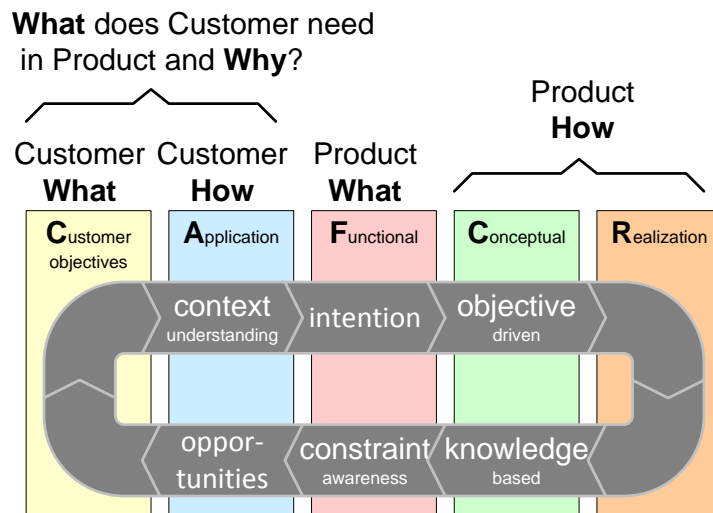


Figure 1.2: Five viewpoints for an architecture. The task of the architect is to integrate all these viewpoints, in order to get a *valuable, usable* and *feasible* product.

figure 1.3 the multiple customers are addressed by applying the CAFCR model recursively.

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

many stakeholders are involved for any given customer. Multiple stakeholders are involved even when the customer is a consumer, such as parents, other family, and friends. Figure 1.4 shows an example of the people involved in a small company. Note that most of these people have different interests with respect to the system.

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

1.4 Life Cycle view

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

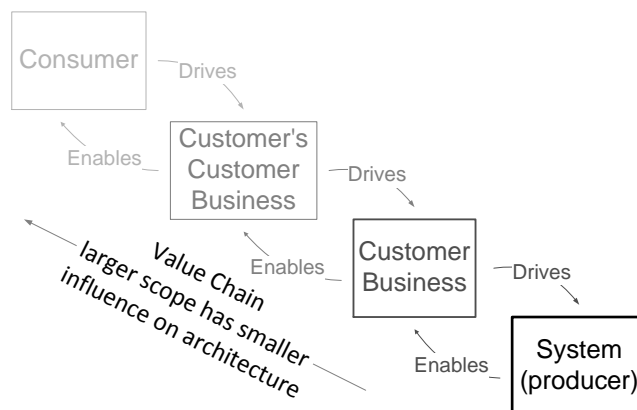


Figure 1.3: CAFCR can be applied recursively

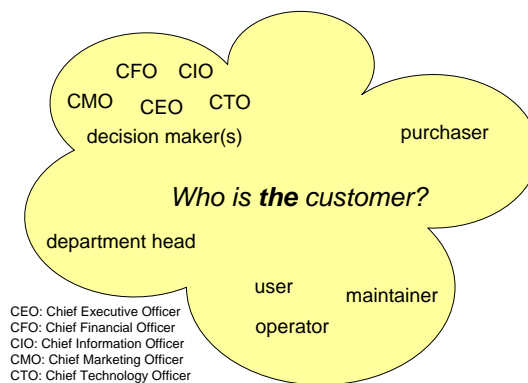


Figure 1.4: Which person is **the** customer?

cycle needs. Figure 1.5 shows the CAFCR+ model that extends the basic CAFCR model with a *Life Cycle view*.

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

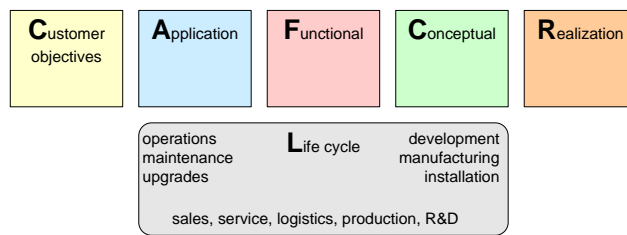
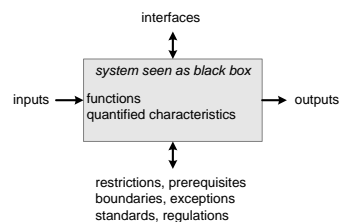


Figure 1.5: CAFCR+ model; Life Cycle View

Chapter 2

Fundamentals of Requirements Engineering



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.

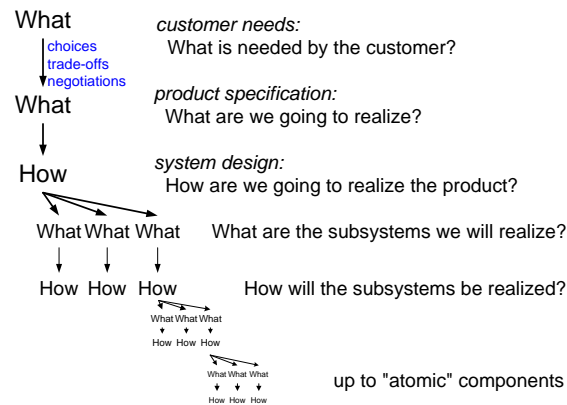


Figure 2.1: The flow of requirements

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.

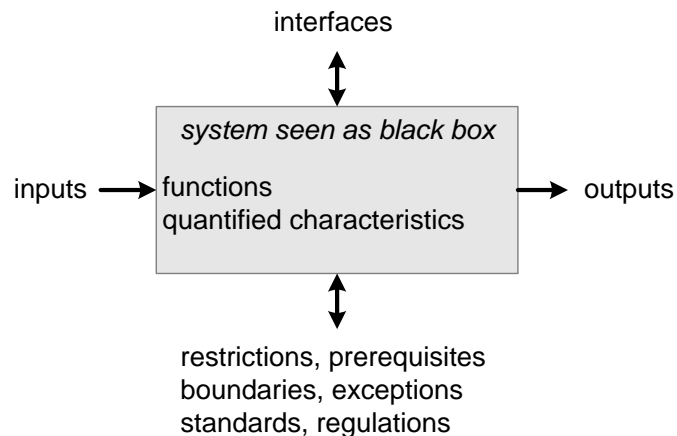


Figure 2.2: System as a Black Box

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

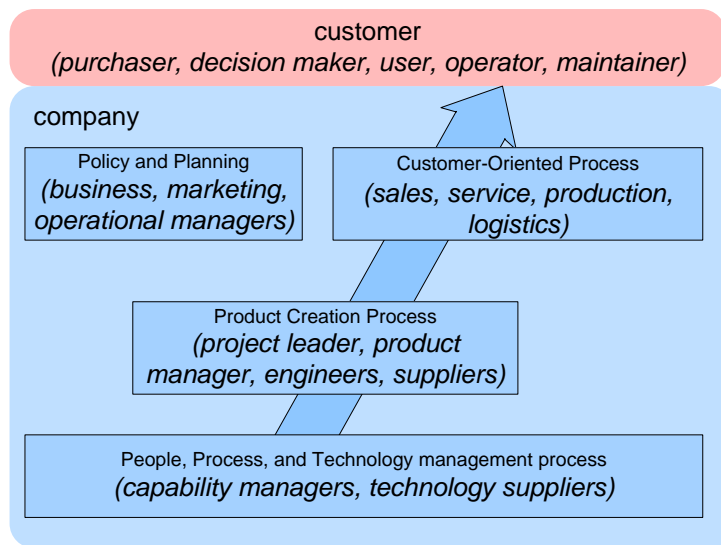


Figure 2.3: A simplified process decomposition of the business. The stakeholders of the requirements are beside the customer self, mainly active in the customer oriented process and the product creation process.

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.

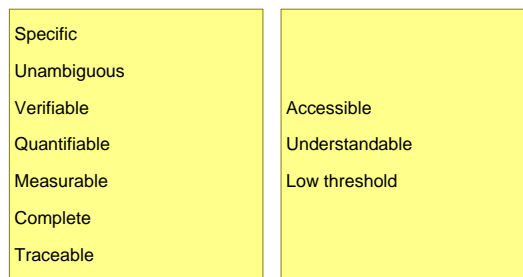


Figure 2.4: Requirements for Requirements

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.

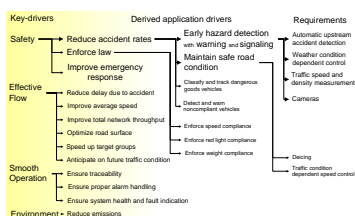
Traceable for all main relations and dependencies. *Traceability* is also a dangerous criterion. Full traceability requires more than infinite time and effort. Understanding how system characteristics contribute to an aggregate performance supports reasoning about changes and decision making.

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.

Chapter 3

Key Drivers How To



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

3.1 Introduction

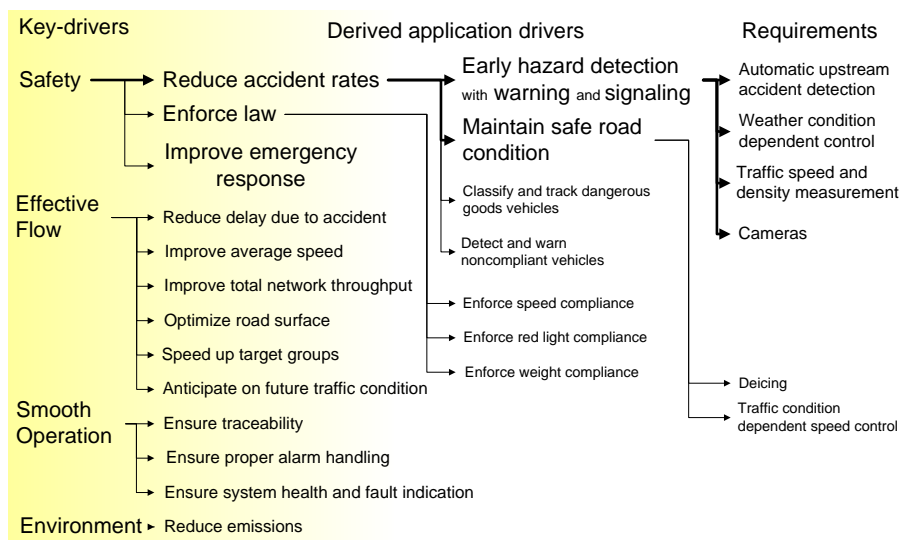
A key driver graph is a graph that relates the key drivers (the essential needs) of the customer with the requirements in the product specification. This graph helps to understand the customer better, and the graph helps to assess the importance of requirements. The combination of customer understanding and value assessment makes the graph into an instrument to lead the project.

We will discuss one example, a Motor way management system, and we will discuss a method to create a customer key driver graph.

3.2 Example Motor Way Management

In this section we discuss an example from practice. The graph discussed here was created in 2000 by a group of marketing managers and systems architects. Creating this version took a few days. Note that we only show and discuss a small part of the entire graph to prevent overload.

Figure 5.2 shows an example of a key driver graph of a motor way management system. A motor way management system is a system that provides information to traffic controllers, and it allows traffic controllers to take measures on the road



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

Figure 3.1: The key driver graph of a Motorway Management System

or to inform drivers on the road. As driver we typically see electronic information and traffic signs that are part of these systems. Also the cameras along the road are part of such system.

The key drivers of a motorway management owner are:

Safety for all people on the road: drivers and road maintainers.

Effective Flow of the traffic.

Smooth Operation of the motorway management.

Environment such as low emissions.

To realize these key drivers the owner applies a number of application processes. For example the traffic controllers can improve safety by reducing the accident rate. The accident rate can be reduced by detecting hazards and warning drivers about the hazards. Examples of hazards are accidents that already have happened and in turn may trigger new accidents. Another example of a hazard are bad weather conditions. Hence the automatic detection of accidents and controls that are weather dependent will help to cope with hazards, and hence will reduce accident rates and improve the safety.

Note that the 4 key drivers shown here are the key drivers of the motor way management system. Other systems will also share these concerns, but might not have these as key drivers. For example, smart phones will have a completely different set of key drivers. Do not use this example as template for your own key driver graph, because it biases the effort.

3.3 CAF-views and Key Drivers

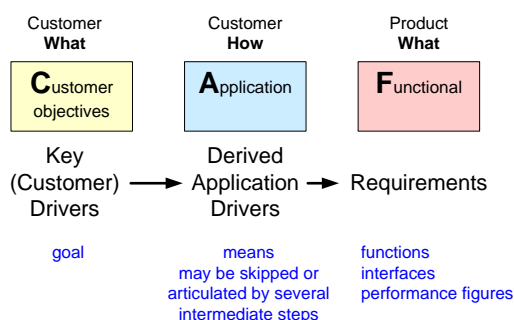


Figure 3.2: The flow from Key Drivers via derived application drivers to requirements

We can capture the essence of the customer world in the *Customer Objectives* view of the CAFCR model by means of customer key drivers. The customer will organize the way of working such that these key drivers are achieved. Figure 3.2 shows how the key drivers as part of the *Customer Objectives* view are supported by application drivers. The application drivers are means to satisfy the customer key drivers. These application drivers in turn will partially be fulfilled by the system-of-interest. Appropriate requirements, e.g. specific functions, interfaces or performance figures, of the system-of-interest will help the customer to use the system to satisfy their customer key drivers. The key drivers are one of the submethods in the Customer Objectives view.

Figure 5.3 shows a method to define key drivers.

Define the scope specific . Identify a specific customer and within the customer a specific stakeholder to make the graph. Choosing a customer implies choosing a market segment. A narrow well defined scope results in a more clear understanding of the customer. The method can be repeated a few times to understand other customers/stakeholders. Products normally have to serve a class of customers. A common pitfall is that the project team too early “averages” the needs and by averaging compromises the value for

• 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: Method to define key drivers

specific customers. We recommend to first create some understanding of the target customers before any compromising takes place.

Acquire and analyze facts We recommend to start building the graph by looking for known facts. For example, in most organizations there is already an extensive draft product specification, with many proposed requirements. For every requirement in the draft specification the *why* question can be asked: “Why does the customer need this feature, what will the customer do with this feature?”. Repeating the *why* question relates the requirement in a few steps to a (potential) key driver.

Note that starting with facts often means working bottom-up¹. When marketing and application managers have a good understanding of the customer, then the facts can also be found in the CA-views, allowing a more top-down approach. Iteration, repeated top-down and bottom-up discussions, is necessary in either case.

Build a graph of relations between drivers and requirements by means of brainstorms and discussions. A great deal of the value of this method is in this discussion, where team members create a shared understanding of the customer and the product specification. Note that the graph is often many-to-many: one requirement can serve multiple key drivers, and one key driver results in many different requirements.

Obtain feedback from customers by showing them the graph and by discussing the graph. Note that it is a good sign when customers dispute the graph, since the graph in that case apparently is understandable. When customers say that the graph is OK, then that is often a bad sign, mostly showing that the customer is polite.

Iterate many times top-down and bottom-up. During these iteration it is quite normal that issues move left to right or opposite due to increased under-

¹Every time that course participants ignore this recommendation, and start top-down while lacking customer insight, they come up with a set of too abstract not usable key drivers.

standing. It is also quite normal that issues are rephrased to sharpen and clarify.

• 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 when defining key drivers

Figure 5.4 shows some recommendations with respect to the definition of key drivers.

Limit the number of key drivers to maximal 6 and minimal 3. A maximum of 6 Key Drivers is recommended to maintain focus on the essence, the name is on purpose **Key driver**. The minimum (three) avoids oversimplification, and it helps to identify the inherent tensions in the customer world. In real life we always have to balance objectives. For example, we have a strong need to maximize safety and performance, while at the same time we will have cost pressure. A good set of key drivers captures also the main tensions from customer perspective.

Do not leave out the obvious key drivers such as the main function of the product. For example, the communication must be recognizable when discussing smart phones; the focus might be on all kinds of innovative features and services, while the main function is forgotten.

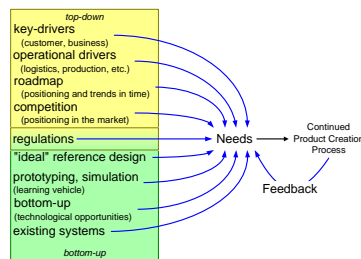
Use short names, recognized by the customer. Key drivers must be expressed in the language of the customer so that customers recognize and understand them. The risk in teams of engineers is that the terminology drifts away and becomes too abstract or too analytical. Another risk is that descriptions or sentences are used in the graph to explain what is meant. These clarifying texts should not be in the graph itself, because the overview function of the graph gets lost. The challenge is to find short labels that resonate with customers.

Use market/customer specific names, no generic names . The more specific a name or label is, the more it helps in understanding. Generic names facilitate the “escape” of diving into the customer world. For example, the term *ease of use* is way too much of a motherhood statement. Instead *minimal number of actions (for experienced users)* might be the real issue.

Allocation to Customer Objectives or Application View Do not worry about the exact boundary between Customer Objective and Application. The purpose of the graph is to get a clear separation of goals and means, where goals and means are recursive: an application driver is a means to achieve the customer key driver, and at the same time it is a goal for the functions of the system of interest. Sometimes we need five steps to relate customer key drivers to requirements, sometimes the relation is obvious and is directly linked. The CAFCR model is a means to think about the architecture, it is not a goal to fit everything right in the different views!

Chapter 4

Requirements Elicitation and Selection



4.1 Introduction

The quality of the system under development depends strongly on the quality of the elicitation process. We can only make a fitting system when we understand the needs of our customer. The outcome of an elicitation process is often an overload of needs. We need a selection process to balance what is needed with all kinds of constraints, such as cost, effort, and time.

4.2 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 4.1 shows a useful number of viewpoints when collecting needs.

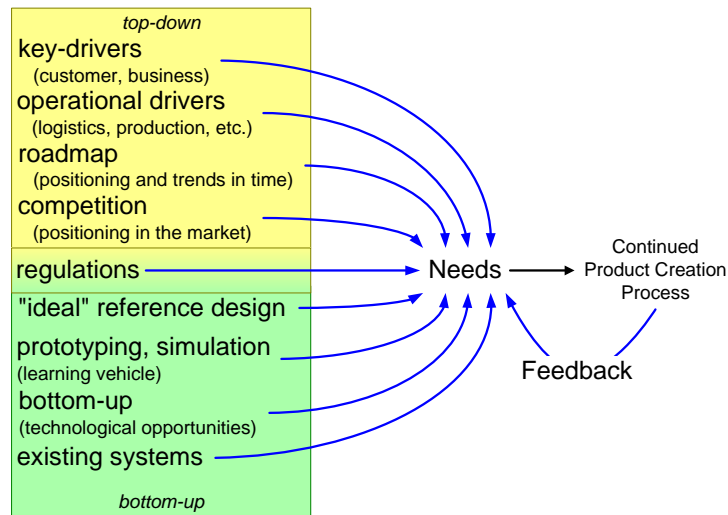


Figure 4.1: Complementary viewpoints to collect needs

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.

4.3 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 4.2 shows the selection process as black box with its inputs and outputs.

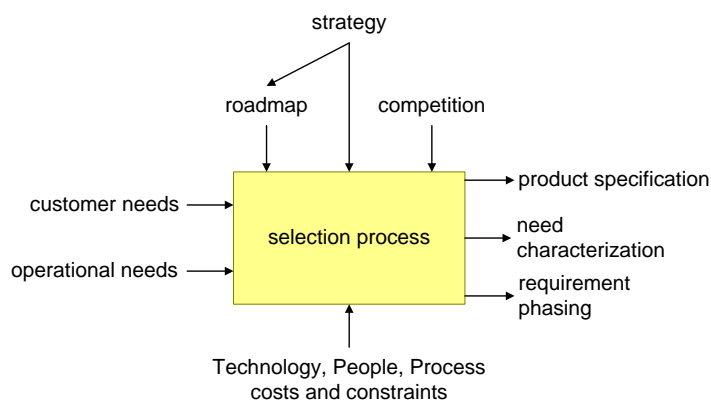


Figure 4.2: The selection process produces a product specification and a phasing and characterization of requirements to prevent repetition of discussion

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 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.

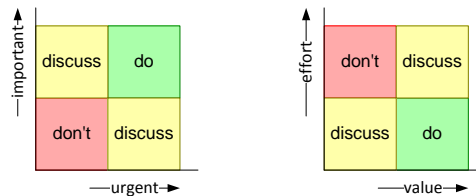


Figure 4.3: Simple methods for a first selection

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 4.3 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
- 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 4.4.

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 [1] a much more elaborated method for requirement evaluation and selection is described.

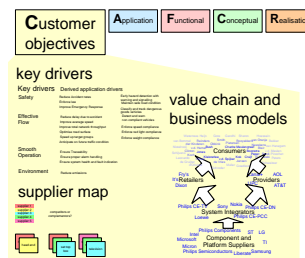
The output of the requirement characterization and the proposed phasing can be used as input for the next update cycle of the roadmap.

- 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 4.4: Quantifiable Aspects for Requirements Selection

Chapter 5

The customer objectives view



5.1 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 5.1 shows an overview of the methods in the customer objectives view.

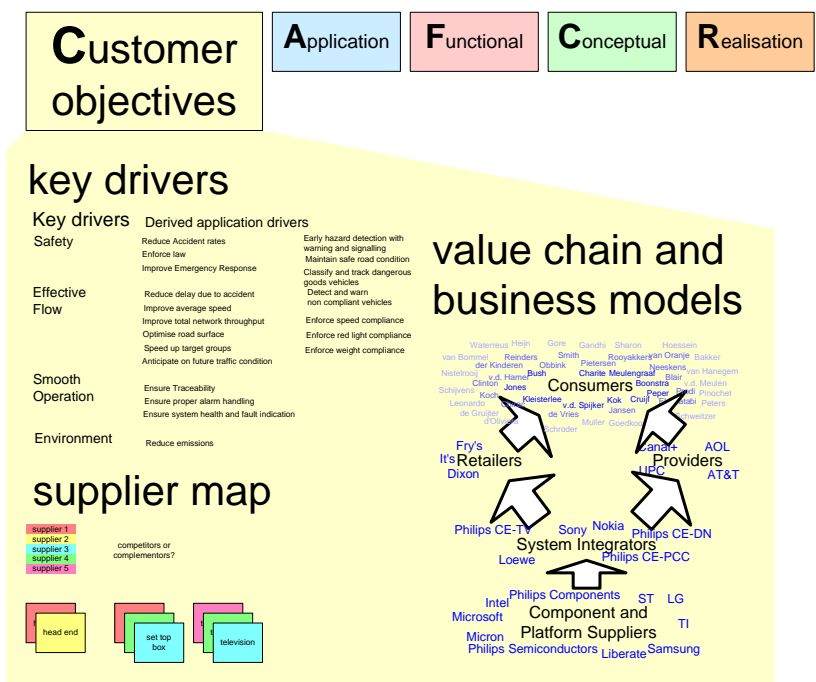


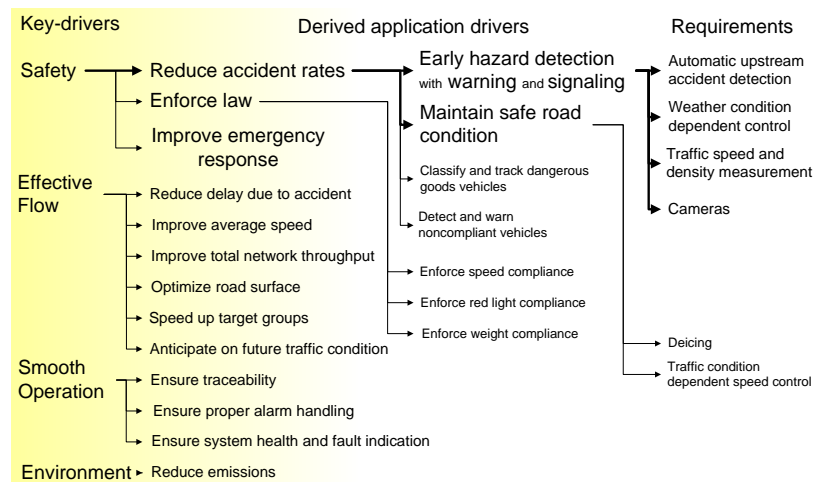
Figure 5.1: Overview of Customer Objectives View methods

5.2 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 5.2 shows an example of key drivers for a motorway management system, an analysis performed at Philips Projects in 1999.

Figure 5.3 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 5.2 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 5.2: 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 5.4 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 5.3: 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 5.4: 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.

5.3 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 5.5 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.

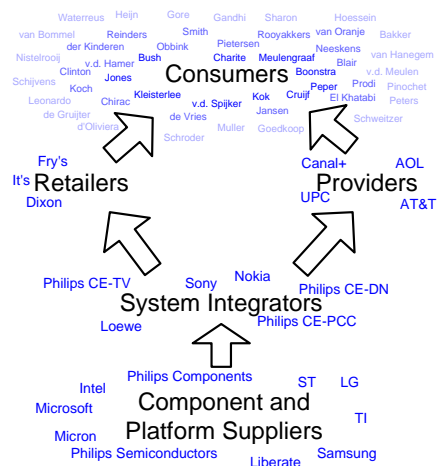


Figure 5.5: Example value chain

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).

5.4 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

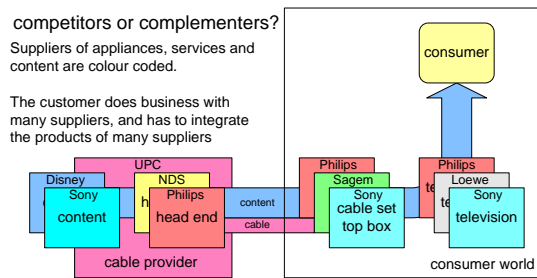
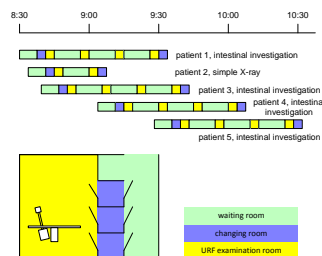


Figure 5.6: Example of simple supplier map for a cable provider

competitors and complementers, which together will supply the solution to the customer. Figure 5.6 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 6

The application view



6.1 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 6.1 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 6.2.

The customer is operating in a wider world, which he only partially controls. A context diagram shows the context of the customer, see section 6.3. 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 6.4) show a static view on the domain, which can be complemented by dynamic models (section 6.5).

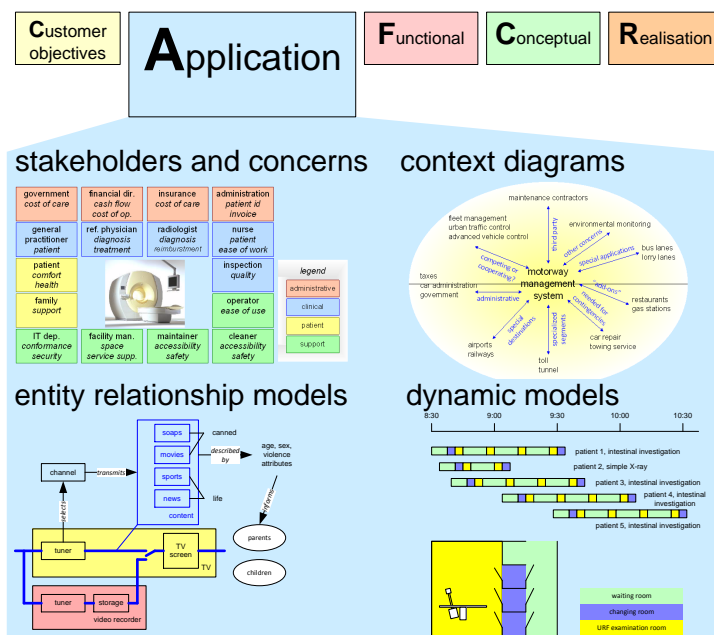


Figure 6.1: Overview of methods and models that can be used in the application view

6.2 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 6.2 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.

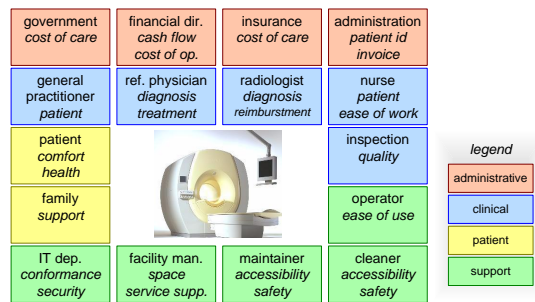


Figure 6.2: Stakeholders and concerns of an MRI scanner

6.3 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.

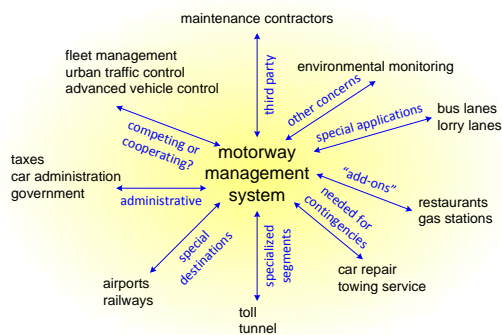


Figure 6.3: Systems in the context of a motorway management system

Figure 6.3 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)

6.4 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 "CAFCE" 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.

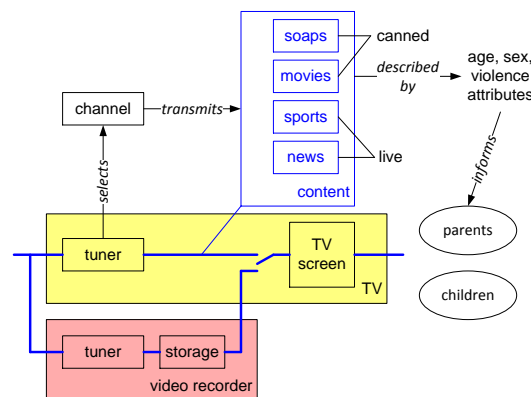


Figure 6.4: Diagram with entities and relationship for a simple TV appliance

Figure 6.4 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.

6.5 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 are of dynamic models are shown in figure 6.5

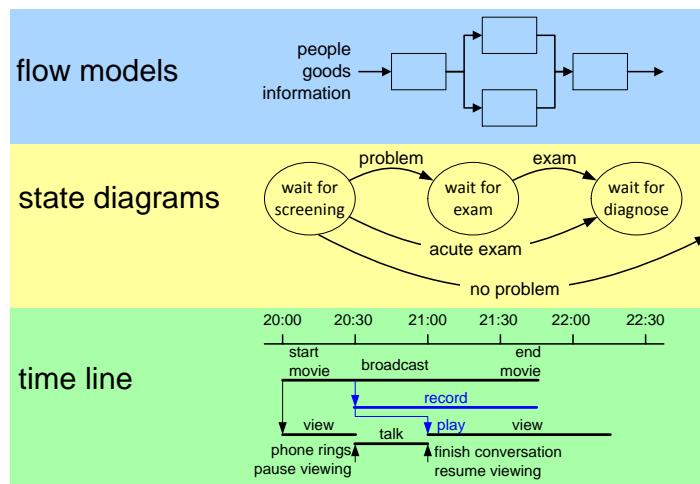


Figure 6.5: Examples of dynamic models

Productivity and Cost of ownership models are internally based on dynamic models, although the result is often a more simplified parameterized model, see figure 6.6.

Figure 6.7 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.

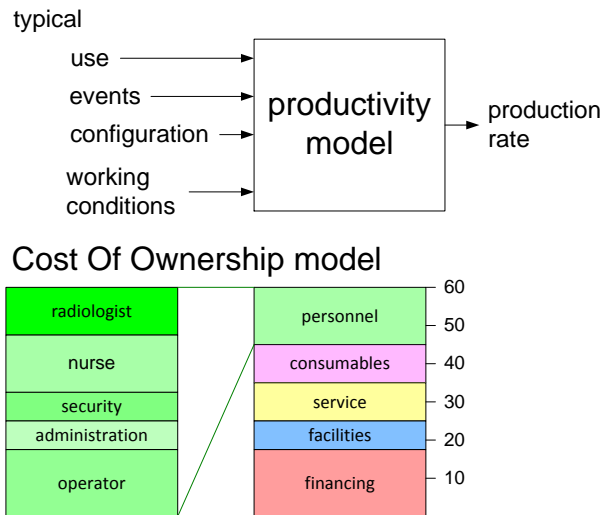


Figure 6.6: Productivity and cost models

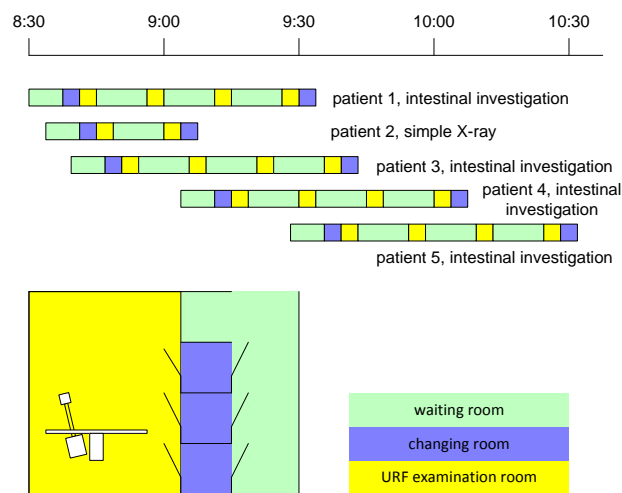


Figure 6.7: Dynamics of an URF examination room

Bibliography

- [1] 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.
- [2] Gerrit Muller. The system architecture homepage. <http://www.gaudisite.nl/index.html>, 1999.
- [3] Henk Obbink, Jürgen Müller, Pierre America, and Rob van Ommering. COPA: A component-oriented platform architecting method for families of software-intensive electronic products. http://www.hitech-projects.com/SAE/COPA/COPA_Tutorial.pdf, 2000.

History

Version: 1.4, date: Septemebr 25, 2014 changed by: Gerrit Muller

- added summary (too much material)

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

- removed CustomerObjectivesView, ApplicationView, ValueChainAndDrivers (too much material)

Version: 1.2, date: June 30, 2010 changed by: Gerrit Muller

- refactored Requirements
- changed status to concept

Version: 1.1, date: October 25, 2004 changed by: Gerrit Muller

- added presentation “Value Chains and Drivers” by Pierre America

Version: 1.0, date: March 25, 2004 changed by: Gerrit Muller

- created reader