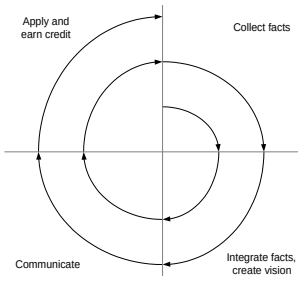
System Architecture



Gerrit Muller
Philips Research IST-SWA-IA
Prof Holstlaan 4 (WL01) 5656 AA Eindhoven The Netherlands
gerrit.muller@philips.com

All Gaudí documents are available at:
http://www.extra.research.philips.com/natlab/sysarch/

version: 0.5 status: preliminary draft 5th October 2001

Contents

List of Figures

List of Tables

Introduction

This book bundles the articles and intermezzo’s 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 [17]. The same information can be found here in presentation format.

Chapters can be read as autonomous units. The sequence choosen 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.

Preface; System Architecture: The Silver Bullet?

0.1 Introduction

The expectation level with respect to processes in general and the system architecture process in particular varies from skeptical to blind faith. The skeptics have experienced that horrible specifications and designs can be pursued under the grand name of Architecture. The followers with blind faith are at the opposite end of the spectrum, their believe in processes inhibits them from seeing the limitations and constraints from the processes applied.

The central message of this Intermezzo is:
Silver Bullets do not exist.

This Intermezzo intends to set realistic expectation levels with respect to the System Architecture Process, and describes the ingredients for successful application.

0.2 Why System Architecture?

System Architecting is a means to create systems efficient and effective, by supplying overview, by guarding consistency and integrity, and by balancing. In other words the System Architect helps the development team to find its way in a rather complex, dynamic and uncertain world.

From psychological point of view people apply their own survival mechanisms, when they perceive a threat. One of the most common survival mechanisms is *The Quest for Certainty*, see subsection 0.2.1.

Unfortunately System Architecting will never remove all uncertainties, see subsection 0.2.2. The application of a system architecture process can help in the risk management, amongst others by prevention, and by minimizing impact.

Successful application of system architecture is far from trivial, section 0.3 describes **how** the System Architecture Process should be applied to meet the goals of efficiency and effectively.

0.2.1 The Quest for Certainty

The majority of people, including managers and engineers, have a need for certainty. Their ideal is to have stable, unchangeable sets of specifications, schedules etcetera. This (hopefully) isolates them from the nasty surprises of reality see table 1.

- incompetent people
- human mistakes
- changing markets
- fast moving competition
- unforeseen physical, chemical, mechanical properties
- mother nature (illnesses, floods)

Table 1: *Nasty Surprises of Reality*

Unfortunately these nasty surprises are a fact of life. Our human capability to control these phenomena is quite limited.

Risk management can help to be more robust. However risk management certainly does not remove these phenomena and it also does not reduce the consequences to zero. Risk management balances probability, effect, and cost.

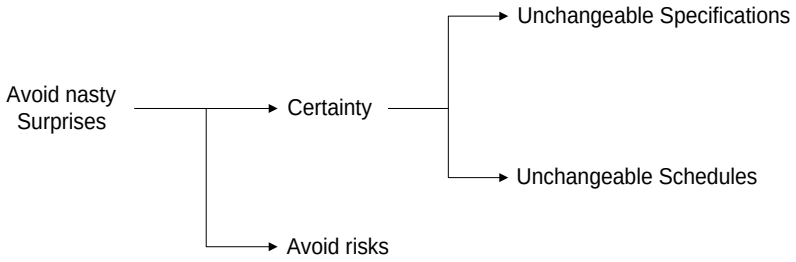


Figure 1: Personal key-driver *to avoid nasty surprises*

People with a need for certainty are willing to accept any method or process which promises certainty. In other words *certainty* appears to be their personal key-driver. It is better to rephrase this key-driver as *to avoid nasty surprises*, which is closer to the internal motivation at the one hand and which gives a handle later on to manage the expectations. Figure 1 visualizes these drivers.

0.2.2 Disclaimer; Setting the Expectations to a realistic level

The Gaudí Project will not deliver a Plug-and-Play System Architecture Process. System architects which have read all the articles and followed the course will not automatically be successful.

The Gaudí project will deliver a large set of consistent background material for system architects. This material ranges from process and architecture principles, providing insight and understanding, to more specific how-to's which provide more directly applicable guidelines.

The competent system architect will use the material by customizing it to the specific problem to be addressed. At the same time the system architect will have to interact with the environment to share this customized way of working.

Whenever the material is applied literal, this is a strong indication that the organization and the system architect do not work explicit enough on the way of working.

0.3 How: Critical Success Factors

Ingredients for an effective application of a system architecture process are shown in table 2.

- Know-How
- Common Sense
- Pragmatics
- Critical attitude
- Drive
- Vision

Table 2: *Critical Success Factors for an effective application of a System Architecture Process*

No method or process will function without these critical success factors. A process can not be used as substitute for know how or common sense.

0.3.1 Know-How

The core of the system architecture work is know-how, ranging from pure technology know-how to application and business know-how. Active control on a broad basis is a prerequisite for a system architect.

0.3.2 Common Sense

Most problems encountered during Product Creation require common sense to solve them. Mechanistic approaches severely limit the solution space, resulting in complex solutions. System architects are capable of "lateral" thinking, allowing solutions in previously unexpected directions.

0.3.3 Pragmatics

The holistic approach can easily derail in a sea of seemingly conflicting requirements and viewpoints. The system architect needs a significant amount of pragmatism to be selective and focused, while being holistic in the back of his mind.

0.3.4 Critical attitude

Clear diagrams, tables with facts and smooth presentations give the impression of high quality and increase the confidence. However these same diagrams, tables and presentations conceal the forgotten, misinterpreted, or underestimated facts. The system architect must always be alert, for instance by asking questions as shown in table 3.

- Do we address the right problem or requirement?
- Is this design adequate?
- Consists the input data from facts, wishes or ideas?
- Do we need so many people for the implementation?
- Does this process or organization fit the problem?

Table 3: *Critical Attitude: Examples of questions to be asked by the System Architect*

0.3.5 Drive

A good system architect has a passion for his architecture, it has an emotional value.

An architect which is working entirely according to the book, obediently going through the motions, will produce a clinical architecture without drive or ownership.

Good architectures have an identity of themselves, which originate in the drive of the architect. Such an architecture is an evolving entity, which is appreciated by the stakeholders.

0.3.6 Vision

The system architect needs to have a vision to be able to provide direction. A vision enables an evolution of existing architectures to desired architectures. Having vision is not trivial, it requires a good understanding of needs (the problem) and means (the solution) plus the trends (opportunities and threats) in both needs and means.

0.4 Summary

The one sentence summary of this intermezzo is: *Silver bullets do not exist.* Table 4 gives a bullet-wise summary.

- Most people want to avoid nasty surprises
- Most people are looking for certainty
- Golden Bullets do not exist
- System Architecture is not a golden bullet
- Critical Success Factors: Know-How, Common Sense, Pragmatics, Critical attitude, Drive and Vision

Table 4: *Summary*

0.5 Acknowledgements

Hans Gieles suggested improvements to increase the cohesion and the red line in this Intermezzo.

Henk Obbink and Angelo Hulshout and many others pointed out that ”The Golden Bullet” should have been ”The Silver Bullet”, which has been changed finally.

Eugene Ivanov pointed out that evolution aspects were missing. The result is the addition of vision as critical success factor.

Chapter 1

The Arisal of a System Architect

1.1 Introduction

System architects are very scarce. This article describes the observed growth pattern of a number of system architects. It is hoped that analysing the characteristics of existing system architects will help in training of new system architects. In [24] a good description is given of a system architect.

1.2 The development of a system architect

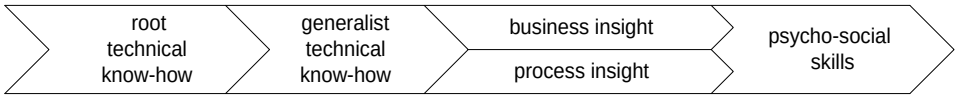


Figure 1.1: Typical Development of a System Architect

System architects need a wide range of know how, skills and experiences to be effective in the job. Figure 1.1 shows a typical development of a system architect.

The root of a system architect is somewhere in the technology. A thorough understanding of a single technological subject is essential. The next step is a broadening of the technical scope. Section 1.3 describes the step from mono disciplinary specialist to multi disciplinary system architect with respect to the technological growth.

When the growing system architect has reached a sufficient level of technological breadth, the discovery is made that most problems have a root cause outside the technology. Two main areas in parallel are opened:

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

- the process side: who is doing what and why

During this phase the architect is broadening mostly in these two dimensions. The view on these dimensions will be rather technocratic. Again when a sufficient level of understanding is reached an awareness starts to grow that people behave much less rational than technical designs. The growing awareness of the psychological and the sociological aspects is the next growing phase.

1.3 Generalist versus Specialist

Most developers of high tech complex products are specialists. They need to have an in-depth understanding of their technological expertise to realize the product development. The decomposition of the development work is most often optimized to create a work breakdown enabling these specialists to do their work as much autonomous as possible.

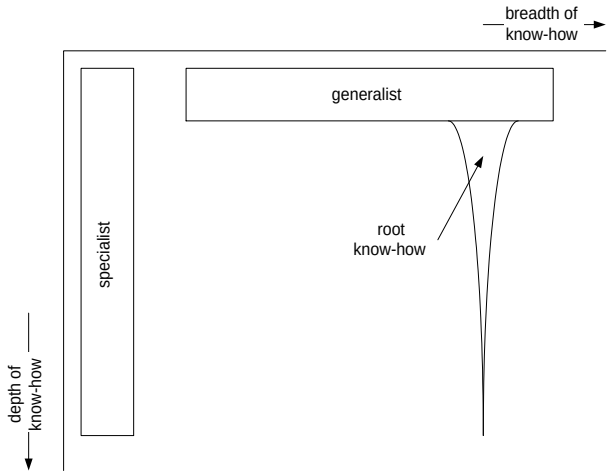


Figure 1.2: Generalist versus Specialist; depth versus breadth

Figure 1.2 shows a visualization of the difference between a specialist and a generalist. Most generalists are constrained in the depth of their knowhow by normal human limitations, such as the amount of available time and the finite capacity of the human mind. The figure also shows that a generalist has somewhere his roots in in depth know how. This root is important for the generalist himself, it provides him with an anchor and a frame of reference. It is also important in the communication with the specialists, because it gives the generalist credibility.

Figure 1.3 shows that both generalists and specialists are needed. The specialists are needed for their in depth knowledge, while the generalists are needed for their more general integrating ability. Normally there are much more specialists required than generalists.

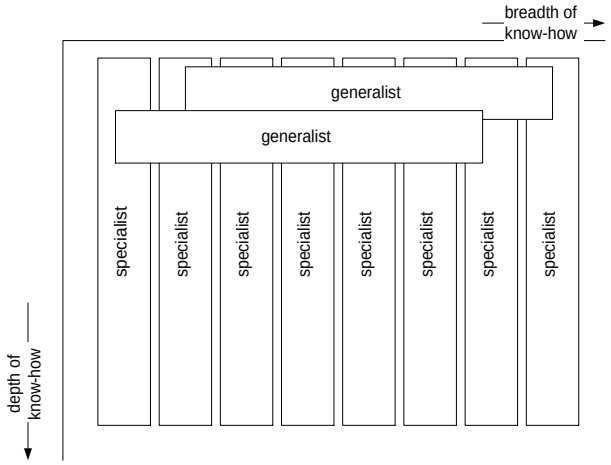


Figure 1.3: Generalists and Specialists are both needed in complex products, they have complementary expertise

There are more functions in the Product Creation Process which benefit from a generalist profile. For instance the function of projectleader or tester both require a broad area of know how.

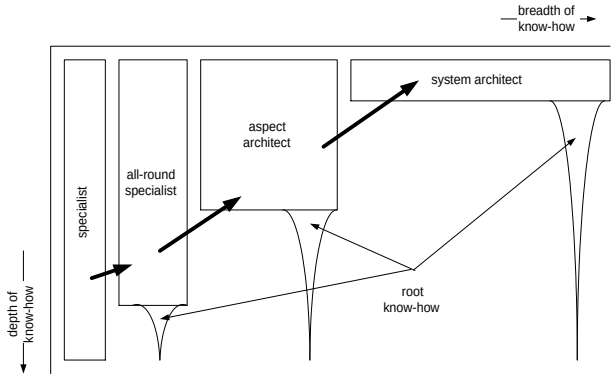


Figure 1.4: Growth in technical breadth, intermediate functions from specialist to system architect

Architects definitely require a generalist profile. One of their primary functions is to generate the integral specification and design of the system. Of course is the step from a specialist to a generalist not a binary transition. Figure 1.4 shows a more gradual spectrum from specialist to system architect. With arrows is indicated that intermediate functions exist in larger product developments, which are natural stepping stones for the arising architect.

Examples of aspect architects are:

- subsystem architects
- SW, mechanics or electronics architects

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

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

Chapter 2

Process Decomposition of a Business

2.1 Introduction

This article positions the system architecture process in a wider business scope. This positioning is intended to help understanding the processes in which the system architect (or team of system architects) is involved.

It focuses on an organization which creates and builds systems consisting of hardware and software. Although other product areas such as solution providers, services, courseware etcetera also need system architects, the process structure will deviate from the structure as presented here.

2.2 Process Decomposition

The business process for an organization which creates and builds systems consisting of hardware and software is decomposed in 4 main processes as shown in figure 10.2.

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

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

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

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

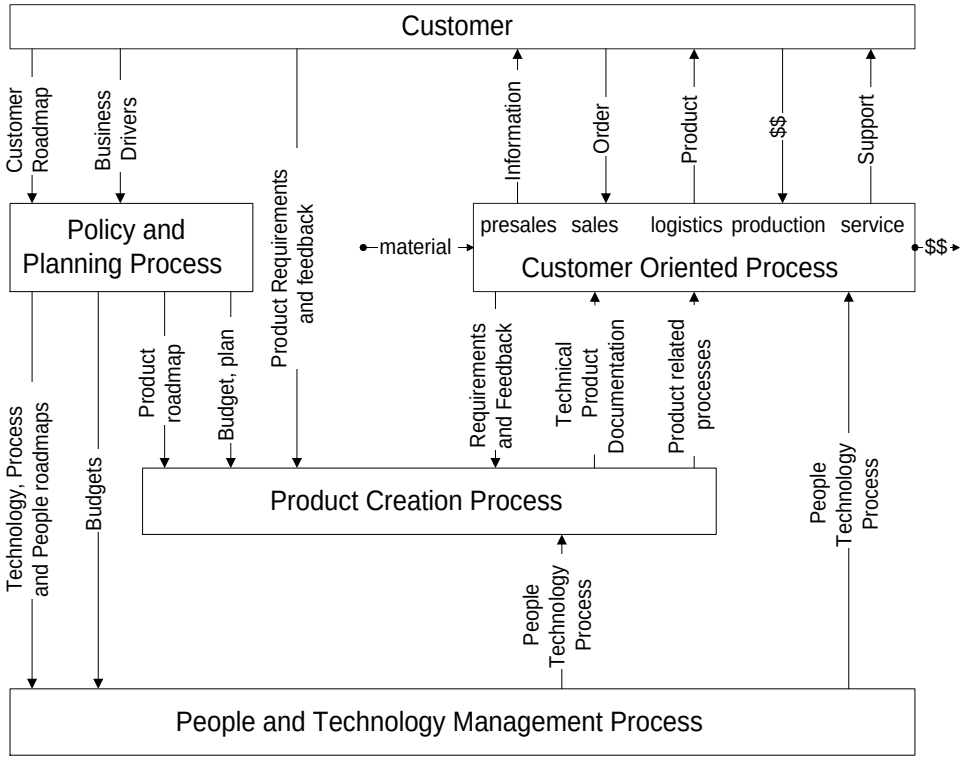


Figure 2.1: Simplified decomposition of the business in 4 main processes

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

The 4 processes as described here are different in nature. The Customer oriented process executes over and over a well defined set of activities. The system architect does not participate in active role in this process. However since the Customer Oriented Process is the main customer of the Product Creation Process, it is imminent that the system architect understands, or better has experienced, the Customer Oriented Process.

In different scopes than the limited scope of organizations which create and builds systems consisting of hardware and software, for instance in solution oriented businesses, the architecture function can be even closer to the customer. This function can be fulfilled by the system architect or by more specialized architects, for instance a solution architect.

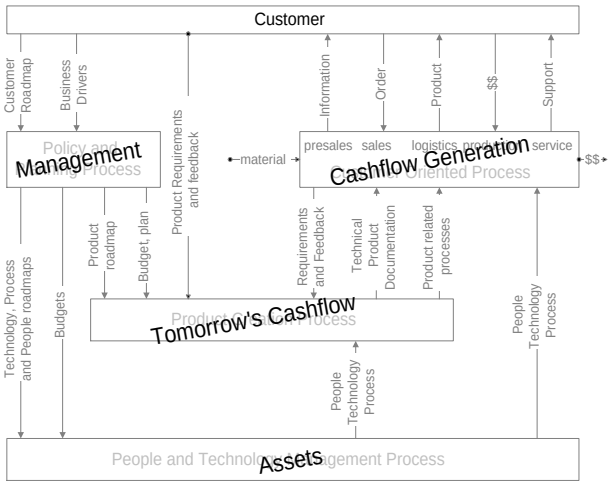


Figure 2.2: Decomposition of the business in 4 main processes, characterized by their financial meaning

The system architect is in continuous interaction with many people, mostly about technical aspects. From this perspective he will generate inputs for the People and Technology Management Process. This might even result in participation in this process for instance by coaching, participation in the appraisal process or participation in technology studies.

The number of instances of each process is related to different entities:

Customer Oriented Process: Depends on geography, customer base, and supply chain.

Product Creation Process: One per entity to be developed, where such an entity can be a product family, a product, or a subsystem.

People and Technology Management Process: One per "competence", where a competence is a cohesive set of technologies and methods.

Policy and Planning Process: One per business. This is the pro-active integrating process.

The split up of the Policy and Planning Process from the Product Creation Process gives the Product Creation Process a clear focus: the entity to be developed.

In this decomposition the evolutionary development of product variants and new releases are seen as individual instances of the Product Creation Process. For example the development of a single new feature for an existing product is performed by following the entire Product Creation Process. Of course some steps in the process will be (nearly) empty, which does not cause any harm.

2.3 Process versus Organization

This process decomposition is not an organization, see [21]. A single person can (and often will) fulfill several roles in different processes.

The system architect specifically will spend most of his time in the product creation (circa. 75%), a considerable amount of time in the policy and planning process (circa 20%) and a small fraction of his time in the people and technology management.

Most engineers will spend a small amount of time in the People Process and Technology Management Process, working on technologies and capabilities, while the majority of their time is spend in the Product Creation Process.

2.4 Value Chain and Feedback

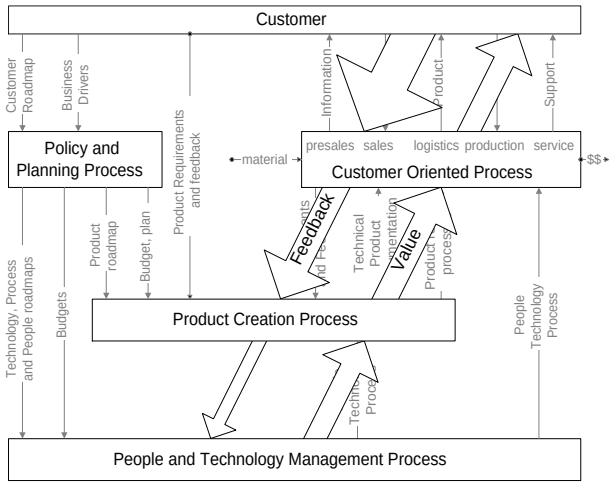


Figure 2.3: The value chain and the opposite feedback flow

The value chain in these processes starts at the People and Technology Management Process, the assets, which is turned into potential money by the Product Creation Process. The Customer Oriented Process finally turns it into real money. Figure 2.3 shows the value chain.

The feedback flows in the opposite direction, from customer via the Customer Oriented Process and the Product Creation Process to the People Technology and Process Management Process.

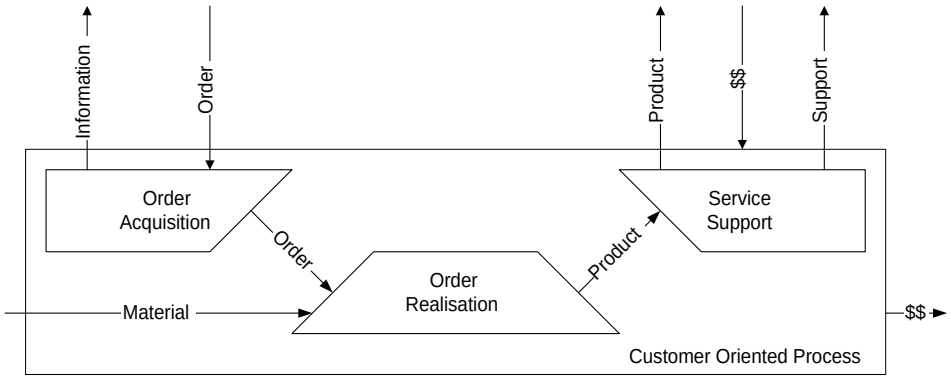


Figure 2.4: Decomposition of the Customer Oriented Process

2.5 Decomposition of the Customer Oriented Process

The Customer Oriented Process is often the largest process in terms of money. From business point of view it is an oversimplification to model this as one monolithic process. Figure 2.4 shows a further decomposition of this process.

The Order Acquisition Process and the Service Support Process are operating quite close to the customer. The Order Realization Process is already somewhat distant from the customer.

The owners of all these 3 processes are stakeholders of the Product Creation Process. Note that these owners have different interests and different characteristics.

2.6 Extended Process Decomposition; Generic Developments

Companies which develop product families try to capitalize on the commonality between the members of the product family. This is often implemented by the development of common subsystems or functions. In the diagram 2.5 this is called **Generic Developments Creation Process**. A wide variety of names is used for this phenomena, such as re-use, standard design, platform etcetera.

2.7 Acknowledgements

Discussions with and critical comments from Rard de Leeuw, Jürgen Müller, Henk Obbink, Ben Pronk and Jan Statius Muller helped to shape, to improve the structure and to sharpen the contents of the article "Positioning the System Architecture

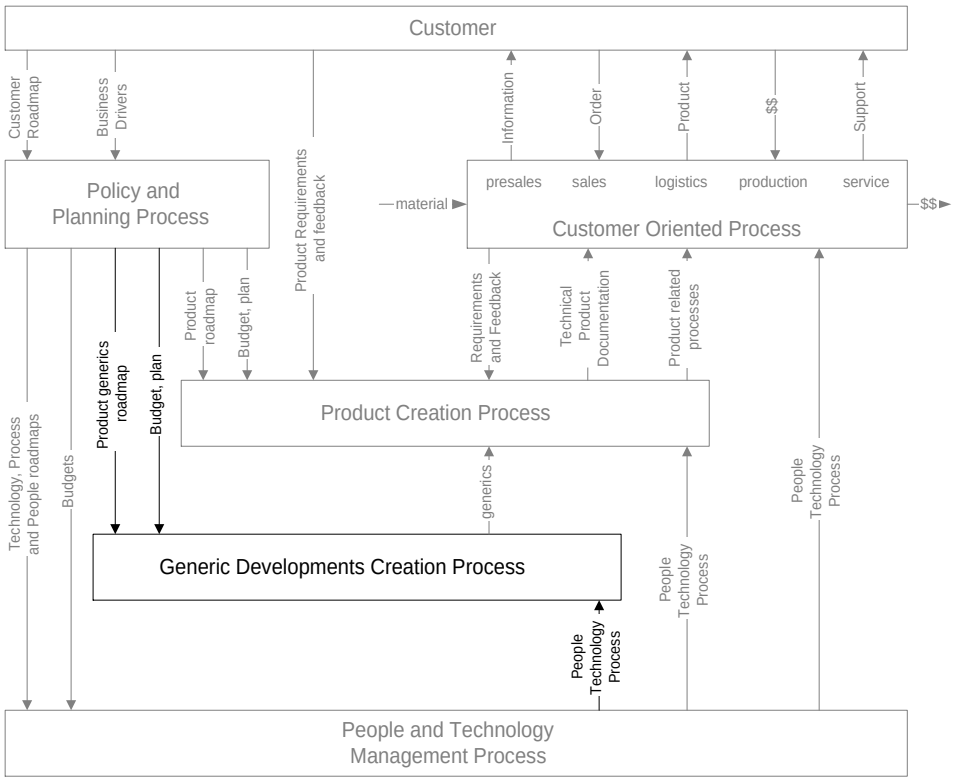


Figure 2.5: The Process Decomposition extended with a generic developments creation process

Process". This intermezzo is based on the first sections of this article. I am grateful for their contribution.

Discussion with Ab Pasman helped to remove some architect bias from the process decomposition, by providing a further decomposition of the Customer Oriented Process.

Jaap van der Heijden helped to improve the layout of the diagrams and with the document structure.

Chapter 3

Intermezzo: What is a Process?

3.1 Introduction

The notion of a process is heavily used in the Gaudí articles. This intermezzo is defining "process" for the context of the Gaudí project, since this word is heavily overloaded. It also discusses the relationship of processes with organizations and the drive for process improvement.

3.2 What is a process

A process in Gaudí context is seen as an abstracted way of working. A process can be characterized by the following attributes:

Purpose What is to be achieved and why

Structure How will the goal be achieved

Rationale What is the reasoning behind this process

Roles Which roles are present, which responsibilities are associated, which incentives are present, what are the criteria for these roles

Ordering Which phasing or sequence is applied

In [8] the following definition is given:

A process is an activity which takes place over time and which has a precise aim regarding the result to be achieved. The concept of a process is hierarchical which means that a process may consist of a partially ordered set of subprocesses.

This definition parallels the characterization above. It adds explicitly the potential hierarchical decomposition of the process itself.

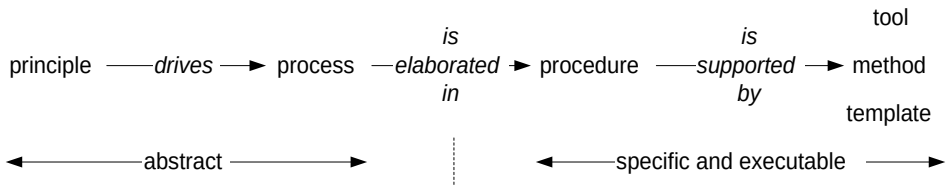


Figure 3.1: A process within an abstraction hierarchy

The notion of a process can be seen as one step in an abstraction hierarchy, as shown in 3.1. The most abstract notion in this hierarchy is the "principle". In the context of the Gaudí project it is planned to explain this notion further in an intermezzo, see also [17].

A process is rather abstract. It describes the essentials of the purpose, structure, rationale, roles and timing, leaving plenty of implementation freedom. The power of a process is its abstraction, which enables its application in a wide range of applications, by tailoring its implementation to the specific application.

A process can be tailored and elaborated in one or more procedures, which describe cookbook-like what need to be done when and by whom. The why in a procedure has often disappeared. The implementation of a procedure is supported by tools, methods, templates and other means.

3.3 The relation between Processes and Organizations

Traditional management is focused on "organizations". Where organization stands for:

- Which functions are needed
- Who is responsible for this function
- What is the hierarchical relation between the functions
- Which meeting structure is required

This management views is insufficient in todays fast moving complex world. The weak spots of this view are:

- Many activities cut arbritarily through the 1-dimensional hierarchy, causing:
  - no ownership, unclear responsibilities

    - high impedance transitions at organizational boundaries
- Functions are a combination of tasks, in most cases no human exists which matches the required skills
- Meeting structures are insufficient to get things done

Processes are more modern instruments for management. Many processes are required to ensure the effective functioning of an organization. These processes are interrelated and overlapping. Processes are non-orthogonal and don't fit in a strict hierarchical structure.

Most complex product developments don't fit in the classical hierarchical organization model, but require a much more dynamic organization model, such as the currently popular chaos based network organization. Processes are the means which help to ensure the output of dynamic organization models such as a chaos based network organization.

Processes can be seen as the blueprint for the behaviour of the people within the organization. People will fullfill multiple roles in multiple processes. The process description is intended to give them an hold on what is expected from them.

All important activities will be covered by a process, requiring the definition of ownership, relation with other processes etcetera. The allocation of roles to people is much more dynamic. This enables a better match between personal capabilities and required skills. In practice this leads to more distribution of responsibilities, making it more feasible to match capabilities and skills.

The 80/20 rule is also valid for processes: 80% of the behaviour is covered by the processes, while 20% requires independent creative behaviour. An organization without processes drowns in chaos, while an organization which blindly implements them will be killed by its own inertia, its inability to adapt to the fast changing world.

For reasons of continuity and stability an hierarchical organization will remain. Tbe least frequently changing dimension is mostly used as a basis for this hierarchy. This hierarchy functions as anchorpoint for people in the continuously changing process world, but should play only a minor role in the entire operation.

The **Centurion** turn around operation within Philips, orchastrated by Jan Timmer in the early nineties, urged the Philips managers and employees to change from an introvert organization point of view to an external result oriented process point of view.

3.4 Process Improvement

Urged by competitive pressure organizations look for ways to improve their efficiency. A huge amount of opportunities for improvement have a strong process component.

The 7S model by McKinsey gives a practical way to improve an organization in a balanced way. The message behind this model is that at least 7 viewsmust be balanced when changing an organization. These 7 views are:

- System
- Structure
- Strategy
- Staff
- Skills
- Style
- Shared Value

The most common pitfall in improvement programs is the over-emphasis on the process component, or worse the isolation of the process improvement. Quite often this happens in organizations which assess their maturity level, for instance by Maturity Models [25].

The Process Improvement Officer [1] is focused on process issues only. Hence where the process view is introduced as an extrovert result oriented appoach, it suddenly turns into an introvert improvement program, where business goals and drivers are unknown.

This is a quite sad situation: The opportunities for improvement are ample with a strong process component, however due to the wrong focus a contraproductive effect is obtained (such as rigid procedures).

Recommendation: Process improvements should originate from the directly involved people, for instance project leaders, engineers, architects etcetera. Invite participation by this group in such a way that they feel the ownership.

3.5 Acknowledgements

Discussions with and critical comments from Rard de Leeuw, Jürgen Müller, Henk Obbink, Ben Pronk and Jan Statius Muller helped to shape, to improve the structure and to sharpen the contents of the article "Positioning the System Architecture Process". This intermezzo is based on the first sections of this article. I am grateful for their contribution.

[1] The existence of this function in itself is quite dangerous, it invites the unbalanced isolated "improvement" behaviour

Chapter 4

The Product Creation Process

4.1 Introduction

The System Architect spends most of his time in the Product Creation Process. This article describes the product creation process, including organizational aspects and the roles of people within the process.

4.2 The Context of the Product Creation Process

Figure 4.1 shows the context of the Product Creation Process in the decomposition of the business in 4 main processes, as described in [19].

From Product Creation Process point of view the Policy and Planning Process determines the framework in which the PCP operates. The Technology and People Management Process supplies people, process and technology enabling the PCP. The customer oriented process is the customer of the PCP.

The Product Creation Process has a much wider context than the classical "Research and Development" or "Development and Engineering" departments. The Product Creation Process includes everything which is needed to create a new product, for instance it includes:

- Development of the production process
- Design of the logistics flow and structure
- Development of required services
- Market announcement
- Market introduction

In other words the Product Creation Process is a synchronized effort of nearly all disciplines within a company.

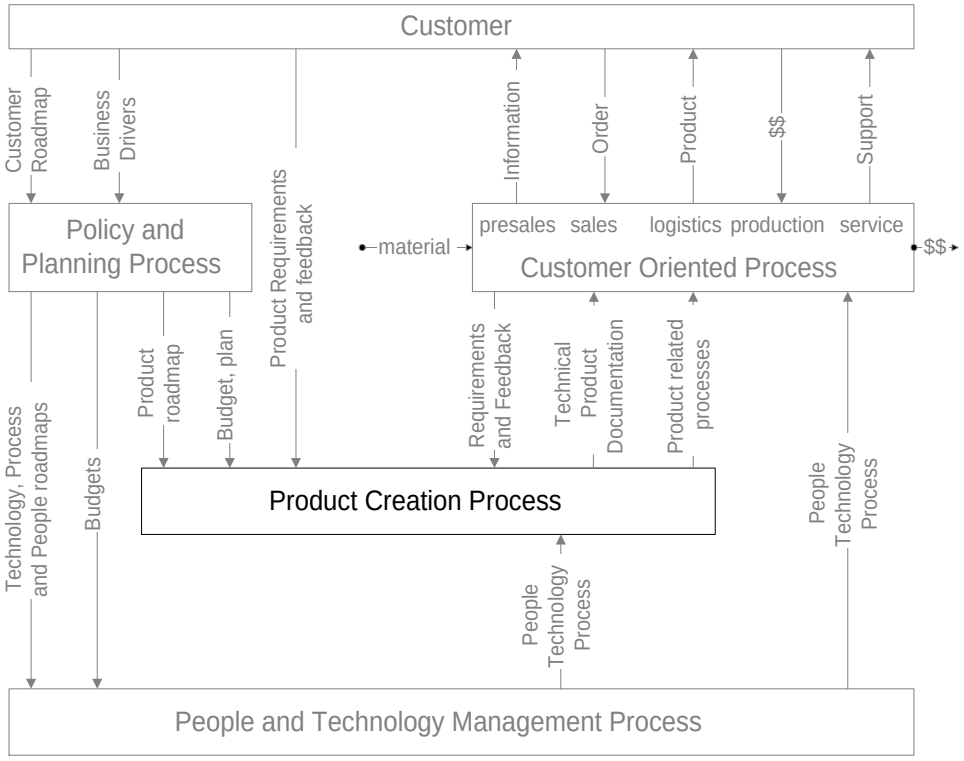


Figure 4.1: Context of the Product Creation Process in the Business

The term Product Creation is not only used for the development of entirely new products, but applies also to the development of variations of existing products or the development of upgrades or add-on products. The implementation of the Product Creation Process can vary, depending on the product being developed; a small add-on product will use a different organization than the development of a large new complex product.

4.3 Phases of the Product Creation Process

The Product Creation Process can be based on a phased approach as a means to structure it. Figure 4.2 shows the phases as used in this article and the participation of all disciplines during this process.

These phases are used across all business functions which have to participate in the Product Creation Process. It is a means to manage the relations between these functions and to synchronize them. Note that sales, production, logistics and service people are involved in the Product Creation Process. Their participation is required to understand the input from the customer oriented process and to help

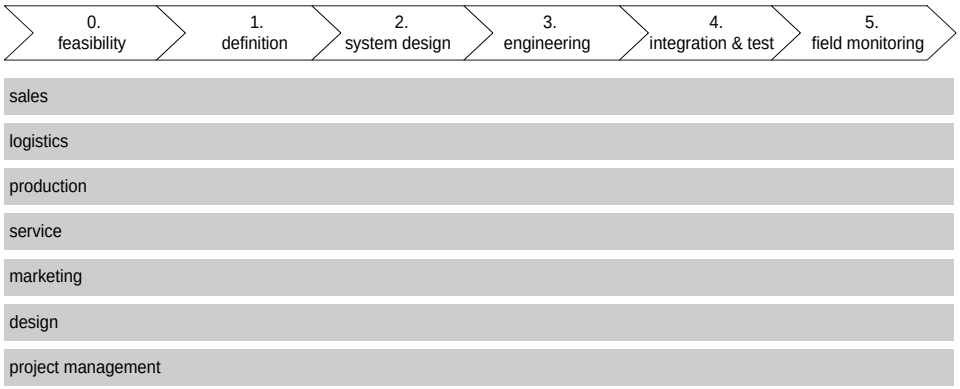


Figure 4.2: A phased approach of the Product Creation Process, showing the participation of all disciplines during the entire process

developing the new processes for the customer oriented process.

Figure 4.3 shows the expected progress for the design deliverables. An analogous phase model is described in [6], which uses only 4 main phases.

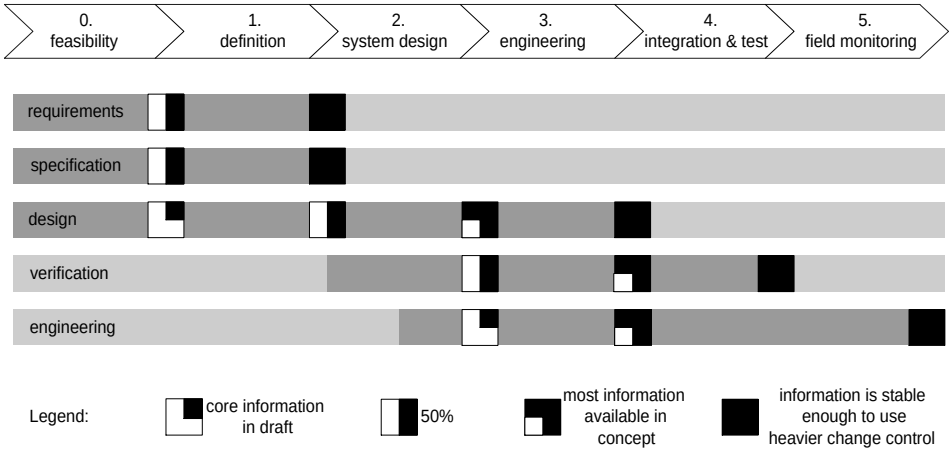


Figure 4.3: A phased approach of the Product Creation Process, showing the progress of the different design deliverables

The advantages of a phased approach are:

- It functions as a blueprint of the way of working.
- The experience of previous projects is re-used.
- People know what is expected from them and when.

- The management team has a reference to judge the status of the project.

The disadvantages of a phased approach are:

- People stop to think and follow the blueprint blindly.
- The implementation of the blueprint is too bureaucratic.
- Phase transitions are misinterpreted as milestones.

It is clear from the above lists of pros and cons of the phased approach that again the common sense is the most important boundary condition for success.

Customization of the phase model to the specific circumstances is always needed. Keep in mind that a phased process is only a means.

The phase process is used as a means for the management team to judge the progress of the Product Creation Process. That can be done by comparing the actual progress with the checklists of the phase model, at the moment of a phase transition. At such a moment the actual progress is measured. Normally the development will continue after the phase review, even if some deliverables are behind schedule. In that case the problem is identified, which enables the project team to take corrective action. Some management teams misinterpret the phase transition as a milestone with mandatory deliverables and disrupt the project by demanding full compliance with the checklist. This kind of interference is very counterproductive. See section 4.4 for a better management method with respect to milestones.

Important characteristics of a phase model are:

inputCharacteristicsPhaseModels

The phase model stresses and supports concurrent activities, see also [6]. A common pitfall is a waterfall interpretation of a phased approach. This can be a very costly mistake, because feedback from implementation and customers is in that case too late in the process. Early and continuous feedback both from implementation as from customer point of view is essential, see [18].

4.4 Milestones and Decisions

A limited number of decisions have to be taken during the development process, which (can) have a large impact on the company. Most of these decisions have to do with a large commitment being made. For instance:

- Ordering of long lead items
- Ordering of expensive materials
- Product announcement

An explicit decision can be planned as a milestone in the project masterplan. To facilitate the decision information should be available, some of the information mandatory to safeguard the company, some of the information only supportive. In general the mandatory information should be minimized to prevent a rigid and bureaucratic process, which will be unresponsive to the outside world.

When the milestone is planned after the phase transition most of the required supportive information will be available in an accessible way.

In summary:

- Define a minimal set of high impact decisions.
- Define the mandatory and supporting information required for the decision.
- Schedule a milestone after the appropriate phase transition.
- Decide explicitly.
- Communicate the decision clearly and widely.

4.5 Organization of the Product Creation Process

The Product Creation Process requires an organizational framework. The organizational framework of the Product Creation Process is independent of the Organizational frameworks of the other processes[1]

4.5.1 Hierarchical decomposition

The operational organization is a dominant organizational view on the Product Creation Process. In most organizations the operations of the Product Creation are decomposed in multiple hierarchical levels, at the highest level the entire product portfolio at the lowest level the smallest operational entity for instance a subsystem. Note that in figure 4.4 the hierarchy stops at subsystem level, although for large developments it can continue into even smaller entities like components or modules. The hierarchy is simply the recursive application of the divide and conquer approach.

The simplification in figure 4.4 is in the assumption that a straight forward decomposition is applied, which is not true when lower level entities are used by different higher level entities. For instance if one subsystem is used in different products. In [14] this aspect is elaborated.

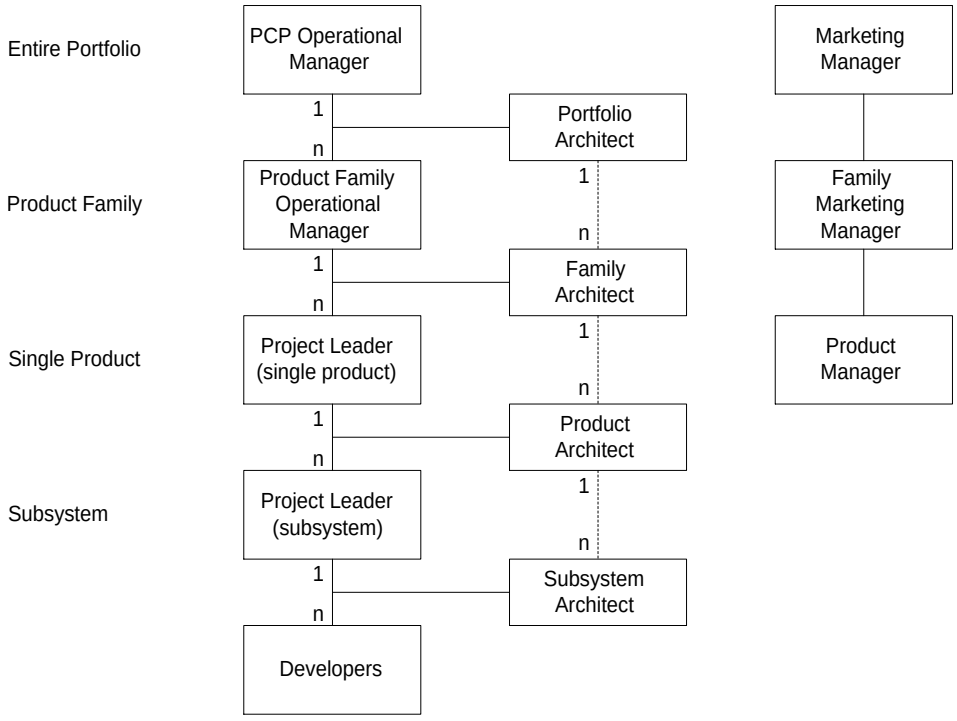


Figure 4.4: The simplified hierarchy of operational entities in the Product Creation Process form the core of the PCP.

4.5.2 Further decomposition of the PCP

The Product Creation Process can be decomposed in 3 processes as shown in 4.5:

Marketing: Defining how to obtain a sellable profitable product, starting with listening to customers, followed by managing the customer expectations, introducing the product at the customer and obtaining customer feedback.

Project Management: Realizing the product in the agreed triangle of

- specification
- resources
- amount of time

Design Control: Specifying and designing the system. The Design Control Process is that part of the PCP which is close to the classical R&D activities. It is the technical part of the PCP.

[1] Quite often a strong link is present between People and Technology Management Process and the PCP; this is quite counterproductive, because these processes have quite different aims and characteristics. Of course nearly all people are part of both organizational frameworks.

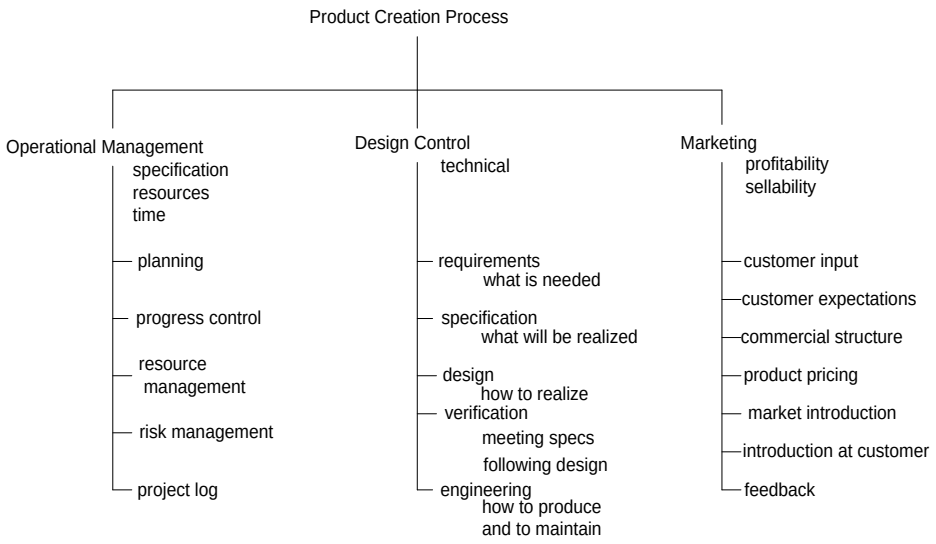


Figure 4.5: Decomposition of the Design Control Process

The functions mentioned in figure 4.4 map directly on the processes in figure 4.5:

- The operational or project-leader is responsible for the operational management
- The architect is responsible for the design control
- The marketing or product manager is responsible for the commercial aspects

4.5.3 Design Control

The ISO 9000 standard has a number of requirements with respect to the "design control" process. The design control process is a hardcore technical process, it is the home base of the system architect. The system architect will support the project management and the commercial process.

The design control process itself is further decomposed, also shown in figure 4.5:

- Requirements
- Specification
- Design
- Engineering
- Verification

The word requirements is quite heavily overloaded. In this context requirements is used to express what the application or user requires of the product, not yet constrained by business or technical considerations. Most development engineers tend to forget the original requirement after several iterations of commercial and technical trade-offs.

The specification describes what will be realized, in terms of functionality and performance. This specification is the agreement with all stakeholders. The difference between the requirements and the specification is that in the specification all trade-offs have been made.

The design is the description how the specification will be realized. For instance the physical and functional decomposition, the budgets for critical technical resources etcetera belong to the design.

Requirements, specification and design are documented in development documents. The main function of these documents is to streamline the Product Creation Process. During this process these are living documents fulfilling an important communication function, while at the same time they play an important role in the control aspect of the design process.

The verification process verifies that the implementation meets the specification in the way it is specified in the design.

The engineering process provides the basis upon which the customer oriented process works for the entire lifecycle of the product. The documentation generated in the engineering process is the output of the Product Creation Process.

4.5.4 Operational Management

The operational management is governed by a simple set of rules:

- An operational leader is asked to perform a job (specification+resources+time).
- If his assessment is that the job is impossible or too risky, then he rejects it.
- Otherwise he accepts and commits to do it.
- Changes of the job require new commitment from the principal and the operational leader.
- The job is executed within the normal (quality) rules of the company.

These rules combine a number of very tightly coupled responsibilities in one function, to enable a dynamic balancing act by the operational leader. These responsibilities form the operational triangle as shown in figure 4.6.

The rules ensure the ownership of the operational leader with respect to the timely delivery of the specification within the agreed budget, with the "standard"

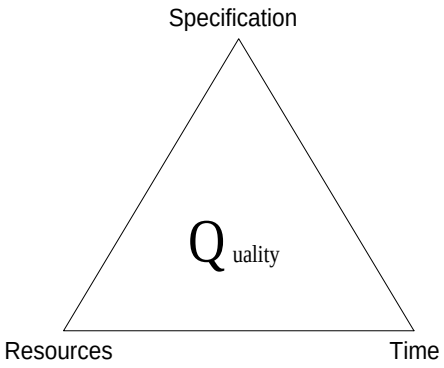


Figure 4.6: The Operational Triangle of responsibilities; The operational leader commits to the timely delivery of the specification within the agreed budget, with the "standard" quality level

quality level. Transfer of one of these responsibilities to another person change the system in an open loop system[2].

In the Product Creation Process more specialized functions can be present. Figure 4.7 shows a number of more specialized functions as part of a number of concentric operational teams. The amount of specialization depends on the size of the operation. In very small developments none of the specializations exist and is even the role of projectleader and architect combined in a single person.

[2]Many conventional development organizations have severe problems with this aspect. The most common mistake is that either the quality responsibility or the resource(budget) responsibility is transferred to the People and Technology Management Process. The effect is that excuses are present for every deviation of the commitment, for instance *I missed the timing because the people were working on non project activities.*

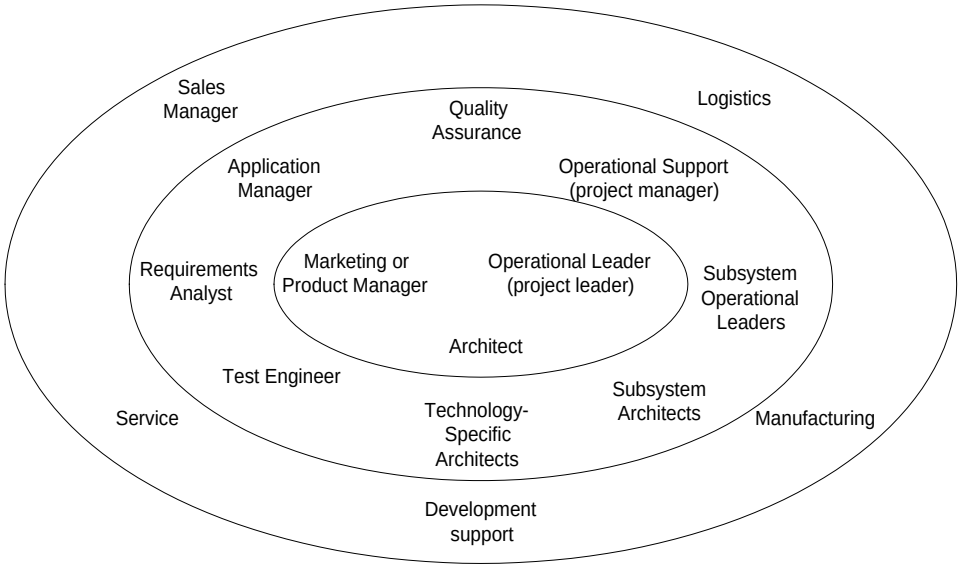


Figure 4.7: The operational teams managing the Product Creation Process

Chapter 5

Intermezzo: The Importance of Feedback for Architecture

5.1 Introduction

Many problems arizing the Product Creation Process could have been prevented or the impact could have been minimized by applying early feedback from "Reality". This Intermezzo describes the Feedback Process as part of the System Architecting Process and explains its importance.

5.2 Why Feedback?

5.2.1 Control

Feedback is used in control systems to ensure that the actual direction corresponds to the desired direction. In general the deviation from the desired direction grows exponentially in time, see figure 5.1.

Many control systems implement a feedback loop to force the system back in the desired direction. Figure 5.1 also shows the effect of a discrete feedback system over time. It will be clear that the sampling interval is determined by the time constant of the deviation and the acceptable deviation level.

Product development can be seen as an ordinary system, which can be controlled analog to technical control systems. Product developments without feedback result in products which are totally out of specification (too late, too slow, too expensive, too heavy etcetera), So any sound development process contains (often multiple) feedback loops.

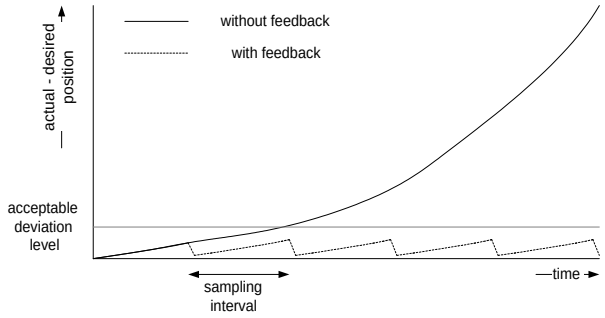


Figure 5.1: The deviation of the actual direction of product development with respect to the desired direction as function of the time

5.2.2 Learning

Human beings learn from their mistakes, *provided that they are aware of them*. Feedback is the starting point of the learning process, because it provides the detection of mistakes. Learning of individuals and of organizations is important to increase efficiency in time, without learning mistakes are repeated, which is a waste of resources.

5.2.3 Applicability

The principle of feedback can be applied on **any** activity. The higher the uncertainty or the larger the duration of an activity is, the more important feedback becomes.

5.3 Theory versus Practice

Many activities performed by the System Architect are by necessity highly theoretical. Some architecture schools promote the system architecture function as strategic, providing direction, without being drowned in operational shit. A second school promotes an architect who is active in the definition phase of a product as well as in the verification phase. The third school of architect, strongly advocated by the author, emphasizes the importance of architecture presence in the entire development lifecycle. In practice many architects function according the fourth school, entirely in the technical domain.

Figure 5.2 visualizes the 4 different schools as function of the process phase. Figure 5.3 shows the amount of theoretical work and the amount of practical work also as function of the process phase.

In this process a number of feedback loops can be closed. Normally the next phase in the process provides feedback to the previous phase in the process. This

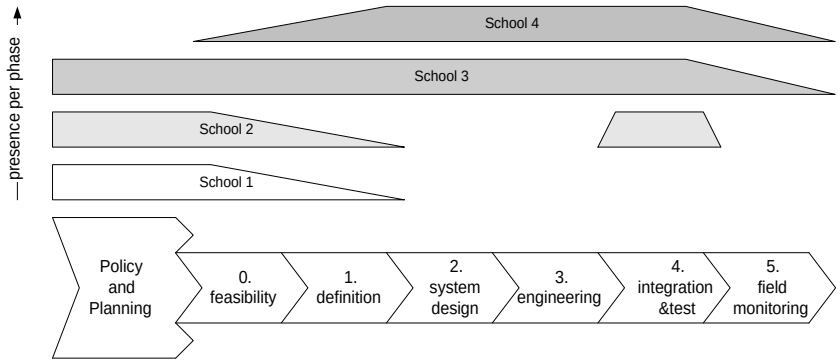


Figure 5.2: 4 Different schools of architecture, showing the presence of the architect in relation to the policy and planning process and the product creation process

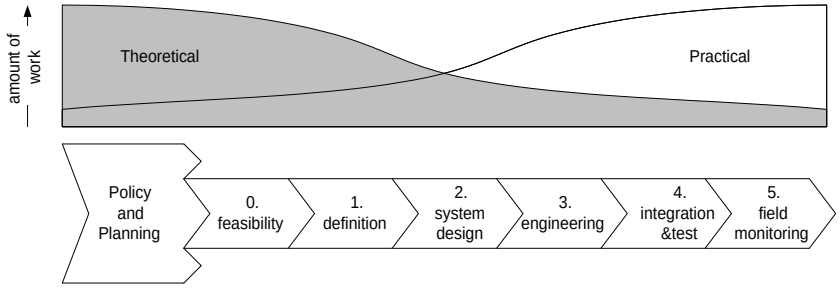


Figure 5.3: Theoretical versus Practical system architecture work in relation to the development lifecycle

feedback nearly always functions correctly, however it is also a rather indirect feedback loop, because the reference is only the usefulness for the next step, while the bottom line reference is the user satisfaction and market success, which cannot be measured by the next step.

The feedback for theoretical work comes from the practical work. Figure 5.4 shows the feedback per development phase. This figure makes it immediately clear that the amount of feedback is proportional to the amount of practical work going on.

5.4 Development Models

The classical V-model or waterfall model is very poor with respect to feedback. More modern processes emphasize the need for more and earlier feedback:

- The Spiral Model

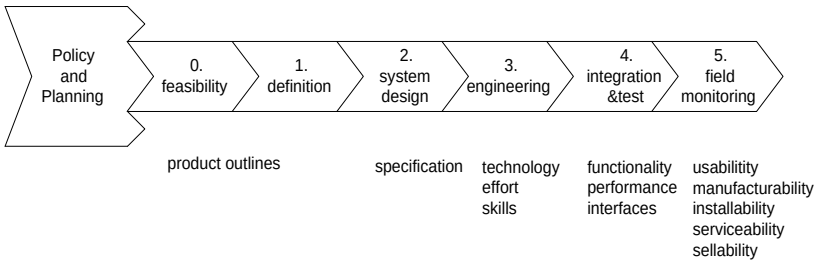


Figure 5.4: Feedback per development phase

- Rational Unified Process
- Extreme Programming
- Open Source

The Spiral Model [2] by Boehm describes how repeated application of a kind of waterfall model in small steps improves the feedback loops.

The Rational Unified Process [6] emphasizes the concurrency of "workflows", with a number of limited (approximately 3) synchronization points per phase as a means of feedback.

Extreme Programming [1] takes the feedback principle to the extreme. The combination of many measures work together to optimize the feedback. This method advocate cycle times of 2 weeks to obtain this.

The Open Source model [23], as far as you can speak of one model, is also based on small steps, with a massive amount of feedback per step.

5.5 Conclusions

The conclusions of this paper are given here as a set of position statements:

1. For the education of a system architect it is essential that he participates in the entire feedback loop.
2. The education of a system architect never stops
3. The system architect for most of his career must participate in the entire product creation lifecycle
4. The value of a system architect in the policy and planning process stems from the practical feedback during the product creation process
5. Feedback can never come too early

6. System architects can have fantastic dreams, feedback is required to prevent that the dream turns into a nightmare

5.6 Acknowledgements

Discussion with Henk Obbink provided valuable input on the topic of feedback.

Chapter 6

The System Architecture Process

6.1 Introduction

System Architecting is being recognized as a critical process in developing complex products, while system architecture skills are scarce.

Currently System Architecting is much of an art, and no clear definition exists for system architecture, while the process of creating, maintaining and evolving a system architecture is also in its early infancy.

This article positions the system architecture process in a wider business scope. This positioning is intended to help understanding the process itself and the role of the system architect (or team of system architects).

It focuses on system architecture within an organization which creates and builds systems consisting of hardware and software. Although other product areas such as solution providers, services, courseware etcetera also need system architects, the process structure will deviate from the structure as presented here.

This article is primarily written for system architects, potential system architects and people which determine the context in which the system architect operates.

An excellent book about system architecture is [24]. The book [10] shows a more mature process for System Engineering. This article fits into a series of articles produced by the Gaudí project as described and partially published in [17].

6.2 System Architecture in the Business Context

In [19] a simplified decomposition of the business is shown. Figure 6.1 shows the main activities of the System Architecture Process as an overlay of the business decomposition.

Processes are goal oriented, see [21]. The process decomposition is not orthogonal, several processes are overlapping. The System Architecture Process is a clear example of this non-orthogonality. Figure 6.2 shows a map of the System Archi-

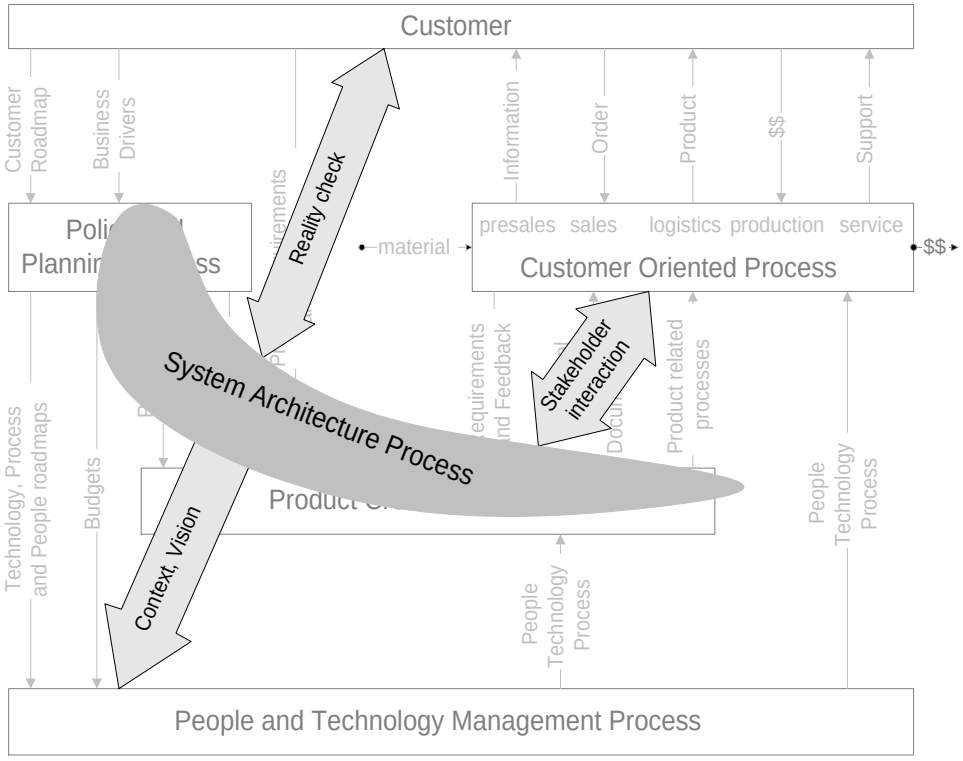


Figure 6.1: The main System Architecture activities in the Business Context

tecture Process and neighboring processes. Many processes, such as manufacturing engineering, service engineering, have been left out of the map, although these processes also have a high architecture relevance.

Both figures make it clear that the System Architecture Process contributes heavily to the Product Creation Process, while it plays also an essential role in the Policy and Planning Process. Both contributions are strongly coupled, see figure 6.3

The System Architecture Process bridges the gap between Product Creation Process and the Policy and Planning Process. In many organizations this link is missing. The absence of this link results in:

- re-inventing a (different) product positioning during the Product Creation Process , with a limited context view
- policies which are severely handicapped by a lack of practicality or realism

The overview created by the System Architecture Process also enables a technology policy.

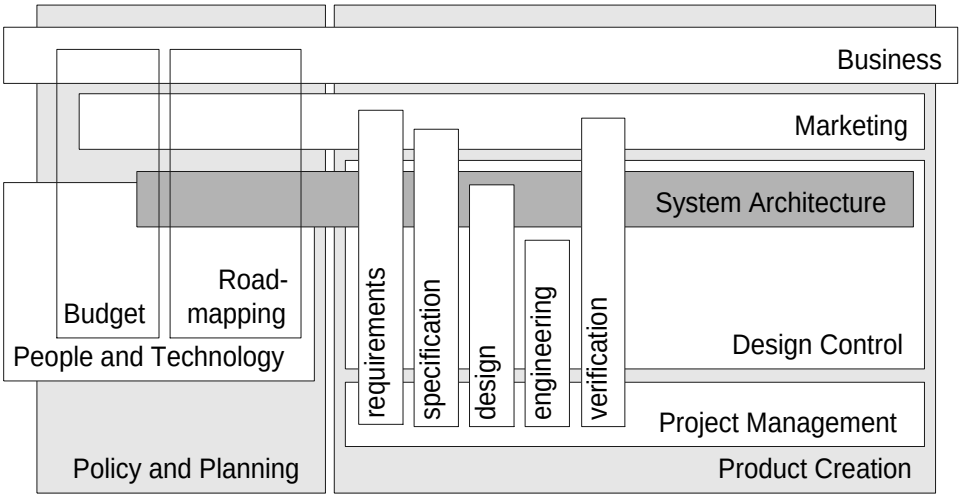


Figure 6.2: Map of the System Architecture Process and neighboring processes

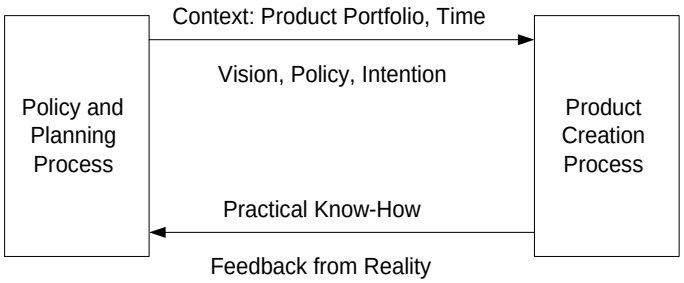


Figure 6.3: Contribution of System Architecture to the the Coupling of Policy and Planning Process and the Product Creation Process

6.3 Purpose of the System Architecture Process

Every business exceeding a few people enables the efficient concurrent work of these people by dividing the tasks in smaller more specialized jobs, the *divide and conquer principle* in action. This decomposition of responsibilities requires an opposing force integrating the activities in a useful overall business result. Several integrating processes are active in parallel, such as project management, commercial management etcetera.

The System Architecture Process is responsible for:

- the Integral Technical aspects of the Product Creation Process, from requirement to deployment.
- the Integral Technical Vision and Synergy in the Policy and Planning Process.

The System Architecture Process is striving for an optimal overall business result, by creating and maintaining the key issues shown in table 6.1

- Balance
- Consistency
- Integrity
- Simplicity
- Elegance

Table 6.1: *System Architecture key issues*

The System Architecture Process is balancing amongst others the goals mentioned in table 6.2.

- External and internal requirements
- Short term needs and long term interests
- Efforts and risks from requirements to verification
- Mutual influence of detailed designs
- Value and costs

Table 6.2: *Goals to be balanced by the System Architecture Process*

Such a balance is obtained by making trade-offs, see for examples of trade-offs table 6.3

It is the purpose of the System Architecture Process to maintain the consistency throughout the entire system, from roadmap and requirement to implementation and verification. On top of this consistency the integrity in time must be ensured.

An enabling factor for an optimal result is *simplicity* of all technical aspects. Any unnecessary complexity is a risk for the final result and lowers the overall efficiency.

Related to *simplicity* is *elegance*. *Elegance* is a rather subjective characteristic [1]. Nevertheless good architectures are often recognized as being elegant.

[1] Elegance is also a very dangerous criterium due to its subjective nature. For example uniformity is by some people equated to elegance, which in many cases is the root cause of a horrible interface or design.

- Performance versus
- Qualities versus
- Functionality
- Synergy versus
- Specific solution

Table 6.3: *Examples of trade-offs made to obtain the right balance*

6.4 The System Architect as Process Owner

The owner of the System Architecture Process is the System Architect or the System Architecture Team. Many other people are involved in the System Architecture Process.

The system architect or the team members spent the majority of their time, about 80%, in the Product Creation Process. From the remaining time the majority is spent in the Policy and Planning Process. In 6.2 it is explained that these processes are strongly coupled. This coupling is for a large part implemented by employing the same people in both processes. A small amount of time is spent in Technology and People Management.

6.5 System Architecture in Product Creation Context

The System Architecture Process is striving for consistency and balance from requirement to actual product. Figure 6.4 shows the high level of concurrency within the Product Creation Process.

The amount of people working in product creation can vary from a few to hundreds [2] of people. All people working on the creation of a new product have only knowledge of a (small) subset of the information. Inconsistencies and local optimal solutions pop up all the time.

The System Architecture Process counteracts this natural degradation of the system quality. Pro-active by clear and sharp requirements, specification and system design as well as reactive by following up the feedback from detailed design, implementation and test.

During the Product Creation Process many specification and design decisions are taken. Quite often these decisions are taken within the scope of that moment, which means that consecutive decisions can be contradictory. For instance a decision

[2] this holds for Philips products, product creation processes which are an order of magnitude larger exist too, for example at Microsoft or Boeing

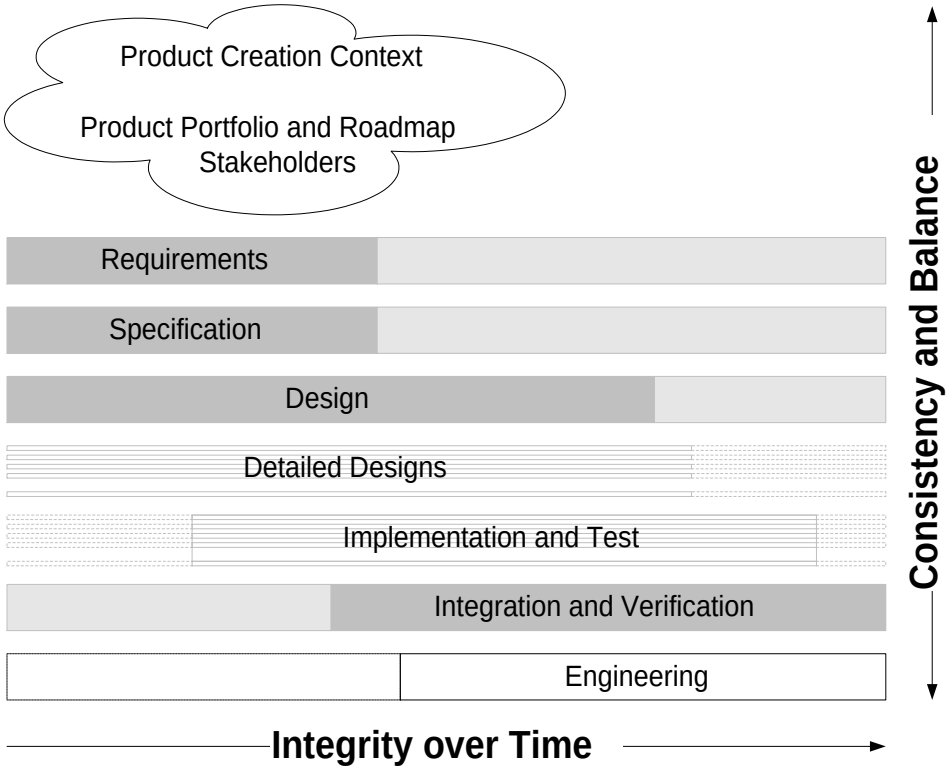


Figure 6.4: System Architecture Activities are highly concurrent in the Product Creation Process

is taken to add memory to the product to increase performance, while one month later the amount of memory is decreased to lower the cost. The System Architecture Process maintains the integrity over time, by looking at decisions from a broader perspective.

6.6 Reference Architecture

A reference architecture abstracts the essential characteristics from 5 different views, see figure 10.4. This abstraction enables it to be used over the entire domain.

The abstraction helps in the Policy and Planning Process to discuss the trends. The detailed reality of the current products can obscure the view on these trends.

In the Product Creation Process the availability of a reference architecture boosts the specification and design process. The System Specification and Design can be focused on the actual performance and critical design issues. The reference architecture functions as a blueprint for the outline specification and design.

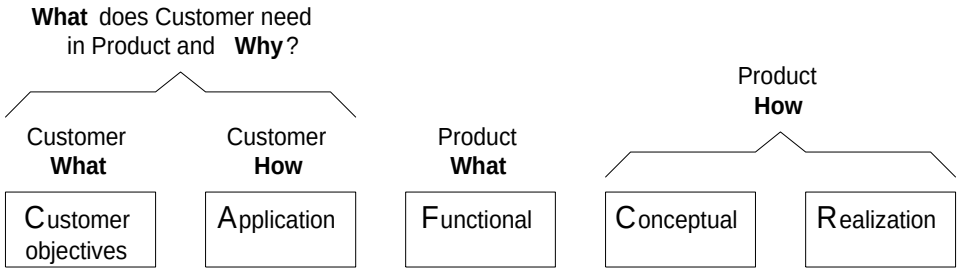


Figure 6.5: A Reference Architecture covers 5 different views

6.7 Acknowledgements

Discussions with and critical comments from Rard de Leeuw, Jürgen Müller, Henk Obbink, Ben Pronk and Jan Statius Muller helped to shape, to improve the structure and to sharpen the contents of the article "Positioning the System Architecture Process". This article is based on the last sections of this article. I am grateful for their contribution.

Jürgen Müller spotted hiccups in the flow of the new article, enabling a streamlining and extension of this article. Robert Deckers analyzed the text and pointed out many inconsistencies and poor formulations.

An inspiring presentation by Bud Lawson helped me to make a more complete and balanced list of System Architecture key issues.

Chapter 7

The Role and Task of the System Architect

7.1 Deliverables of the System Architect

The deliverables of a System Architect are stacks of paper, or the electronic equivalent, symbolized by the stack in figurefig:RoleSAdeliverables.

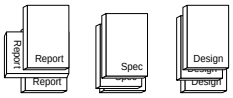


Figure 7.1: Deliverables of a system architect consists of a stack of paper

Table 7.1 shows the main deliverables of a System Architect. Quite often the System Architect does not even produce all deliverables mentioned here, but does he take the responsibility for these deliverables by coordinating and integrating contributions of others.

7.2 System Architect Responsibilities

The System Architect has a limited set of primary responsibilities, as visualized in figure 7.2. The system architect has many secondary responsibilities, which are more specific. These secondary responsibilities have an owner, as shown in table 7.2.

In [20] the purpose of the system architecture process is described in the same terms as used here. In short the primary responsibility of the System Architect is to ensure the good functioning of the System Architecture Process. In practice this responsibility is often shared by a team of System Architects, with one chief architect taking the overall responsibility.

- Requirements (**what** is needed)
- Specification (**what** will be realized)
- Design (**how** the system will be realized)
- Verification Specification (**how** the system will be verified)
- Verification Report (the result of the verification)
- Feasibility Report (the results of a feasibility study)
- Roadmap

Table 7.1: *Classification of the main deliverables of a System Architect*

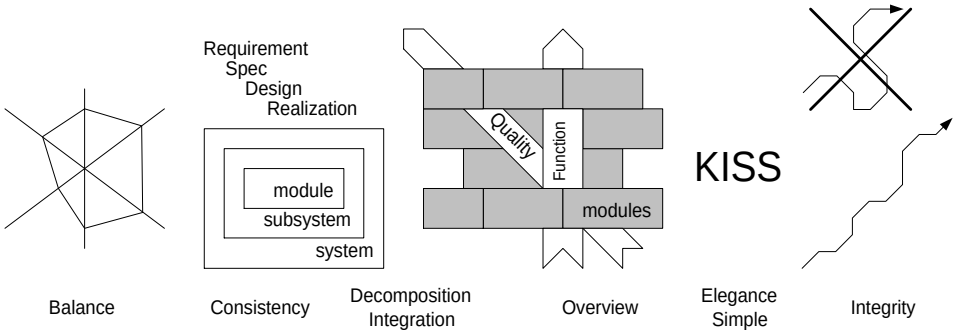


Figure 7.2: The primary responsibilities of the system architect are not "SMART"

7.3 What does the System Architect do?

Figure 7.3 shows the variety of activities of the day to day work of a system architect. A large amount of time is spent in gathering, filtering, processing and discussing detailed data in an informal setting. These activities are complemented by more formal activities like meetings, visits, reviews etcetera.

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

Abstractions only exist for concrete facts

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

Figure 7.4 shows the bottom up elicitation of higher level views. A system architect sees a tremendous amount of details, most of these details are skipped, a smaller amount is analyzed or discussed. A small subset of these discussed details

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

Table 7.2: *(Incomplete) list of secondary responsibilities of the system architect and the related primary owner*

is shared as an issue with a broader team of designers and architects. Finally the system architect consolidates the outcome in a limited set of views. The order of magnitude numbers cover the activities in one year.

The opposite flow in 7.4 is the implementation of many of the responsibilities of the system architect. By providing overview, insight and fact-based direction a simple, elegant, balanced and consistent design will crystalize, where the integrity of designs goals and solutions are maintained during the project.

7.4 Task versus Role

The task of the system architect is to generate the agreed deliverables, see section 7.1 This measurable output is requested and tracked by the related managers: the project leader and the line manager. Many managers appreciate their architects only for this visible subset of their work.

The deliverables are only one of the means to fulfil the System Architect Responsibilities, as described in section 7.2. The system architect is doing a lot of nearly invisible work to achieve the system level goals, his primary responsibility. This work is described in section 7.3. Figure 7.5 shows this as a pyramid or iceberg: the top is clearly visible, the majority of the work is hidden in the bottom.

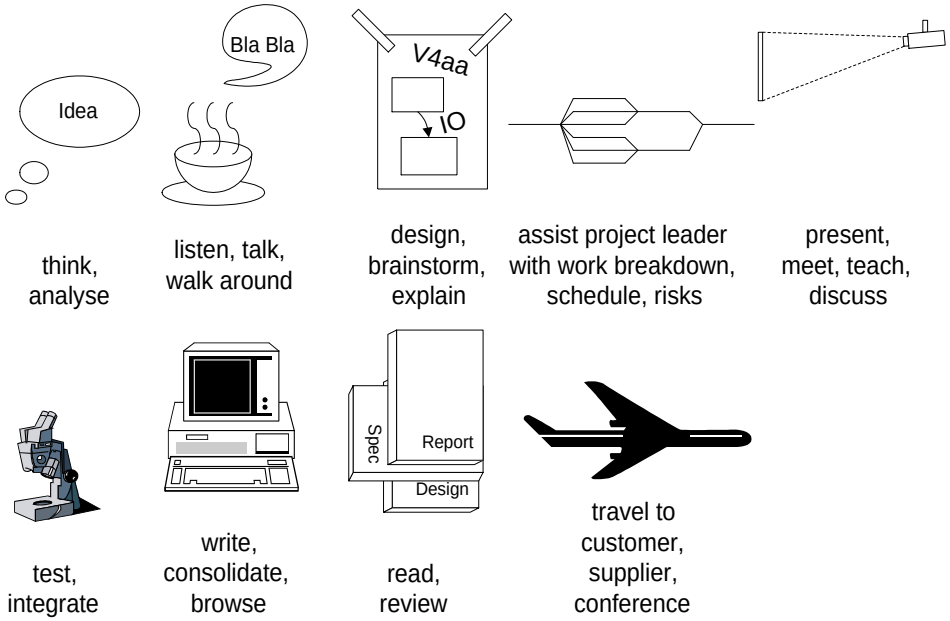


Figure 7.3: The System Architect performs a large amount of activities, where most of the activities are barely visible for the environment, but which are crucial for his functioning

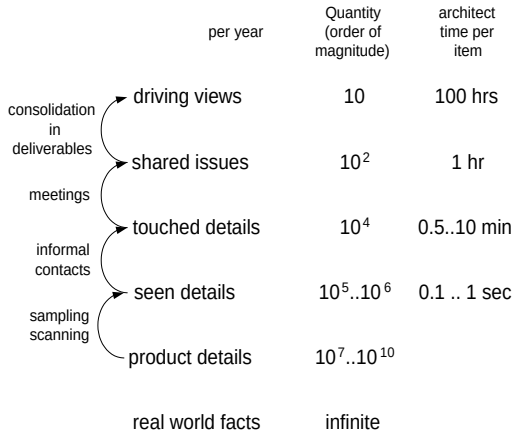


Figure 7.4: Bottom up elicitation of high level views

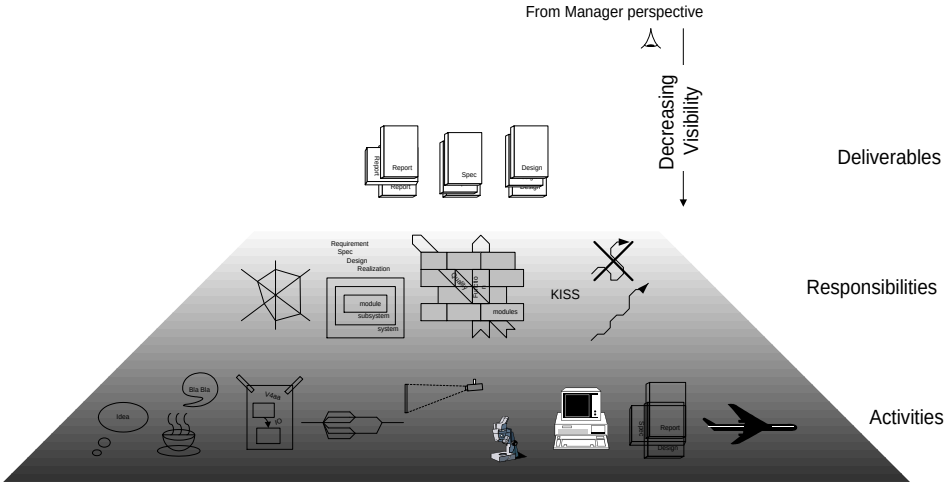


Figure 7.5: The visible outputs versus the (nearly) invisible work at the bottom

Chapter 8

Function Profiles; The sheep with 7 legs

8.1 Introduction

Which people have the potential to become good system architects? How to select (potential) system architects? Many human resource and line managers struggle with these questions.

This article lists a number of characteristics of individuals and maps the relative importance of these characteristics for different jobs, such as system architect, developer and line manager.

8.2 System Architect Profile

The profile of the ”ideal” system architect shows a broad spectrum of required skills. Quite some emphasis in the skill set is on *interpersonal skills*, *know-how*, and *reasoning power*.

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

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

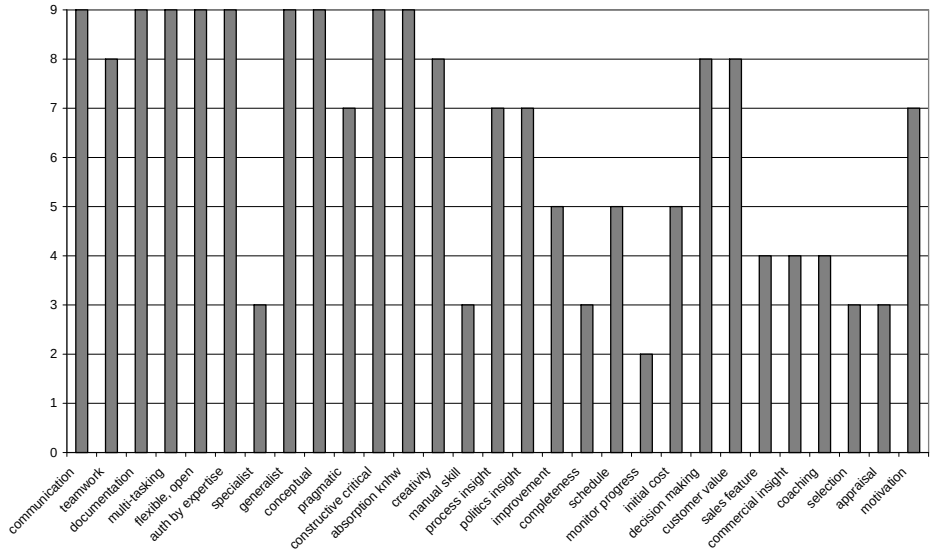


Figure 8.1: The function profile of the system architect

8.2.1 Most discriminating characteristics

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

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

The first reduction step is to select the generalists only, which reduces the input stream with one order of magnitude.

The next step is to detect those people which need time and concentration to make progress. These people become unnerved in the job of the system architect, where frequent interrupts (meetings, telephone calls, people walking in) occur all the time. Ignoring these interrupts is not recommendable, this would block the progress of many other people. Whenever these people become system architect nevertheless they are in sever danger of stress and burn out, hence it is also the benefit of the person itself to fairly asses the multi-tasking characteristic.

The attitude of the (potential) architect is important for the long term effectiveness. Architects who work on the basis of delegated power are often successful

on the short term, creating a single focus in the beginning. However in the long run the inbreeding of ideas takes its toll. Architecting based on know-how and contribution costs a lot of energy, but it pays back in the long term.

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

8.3 Test Engineer Profile

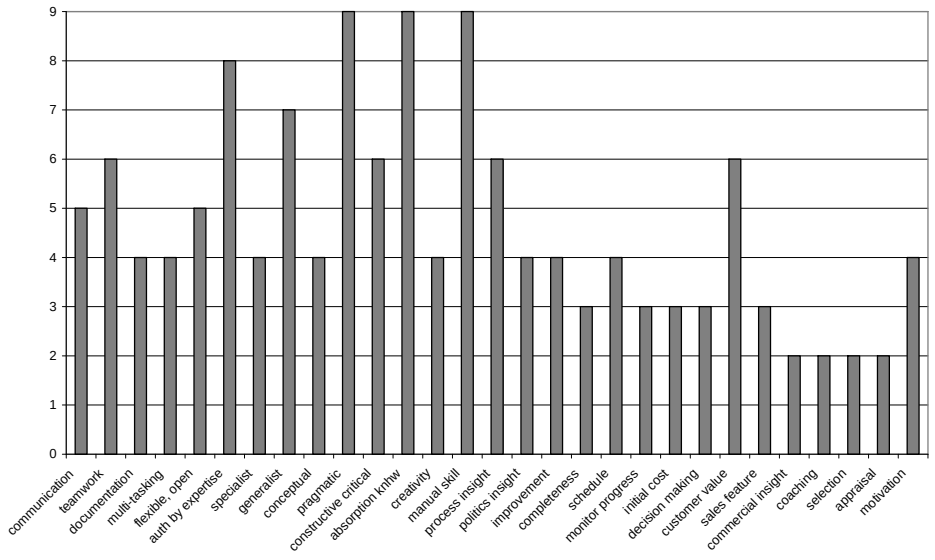


Figure 8.2: The function profile of the test engineer

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

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

8.4 Developer Profile

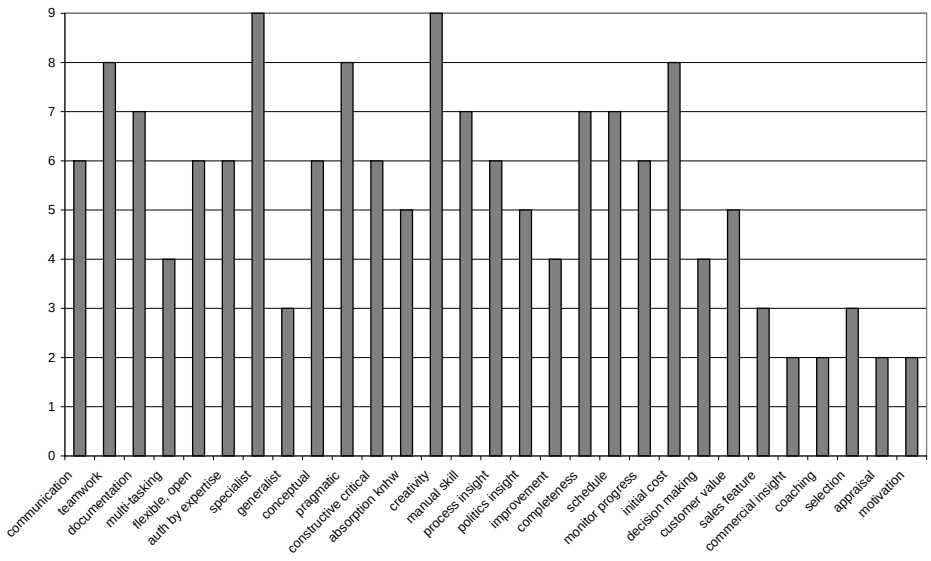


Figure 8.3: The function profile of the developer

The core value of the developer is his specific know-how. A good developer excels in a limited set of specialisms, knowing all tricks of the trade. On top of this he should be able to deploy this knowhow in a creative way. In today's large development teams a reasonable amount of interpersonal skills are reuired as well as reasoning power and project management skills.

8.5 Operational Leader Profile

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

The multi-tasking ability is an important prerequisite for the operational leader. If this ability is missing the person runs a severe risk on a burn out.

8.6 Line Manager Profile

The line manager manages the intangible assets of an organization: the people, the technology and the processes. Technology and process know-how are tightly coupled with people, this know-how largely resides in people and is deployed by

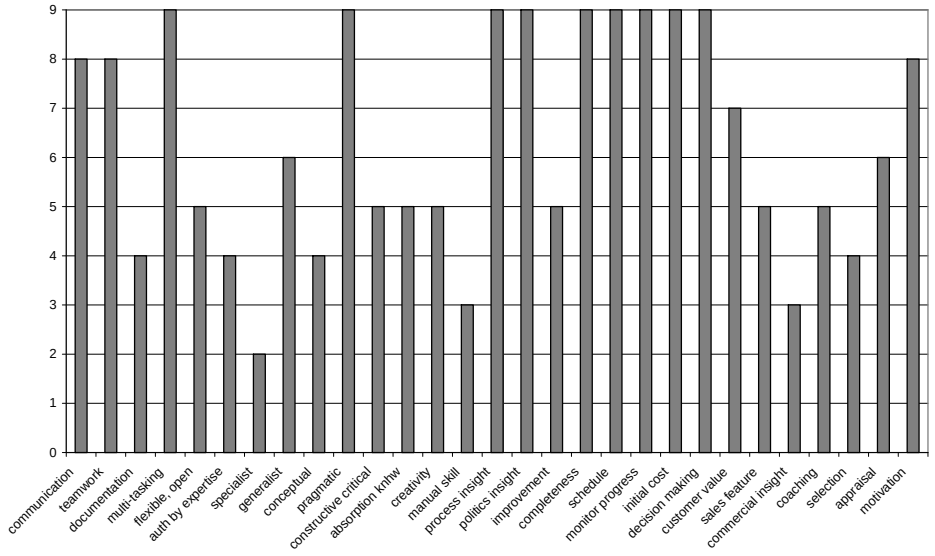


Figure 8.4: The function profile of the operational leader

people. Human resource management skills and process skills are the core discipline for line managers, which need to be substantiated with sufficient *specialist* know-how.

8.7 Commercial Manager Profile

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

8.8 Definition of Characteristics

8.8.1 Interpersonal skills

communication

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

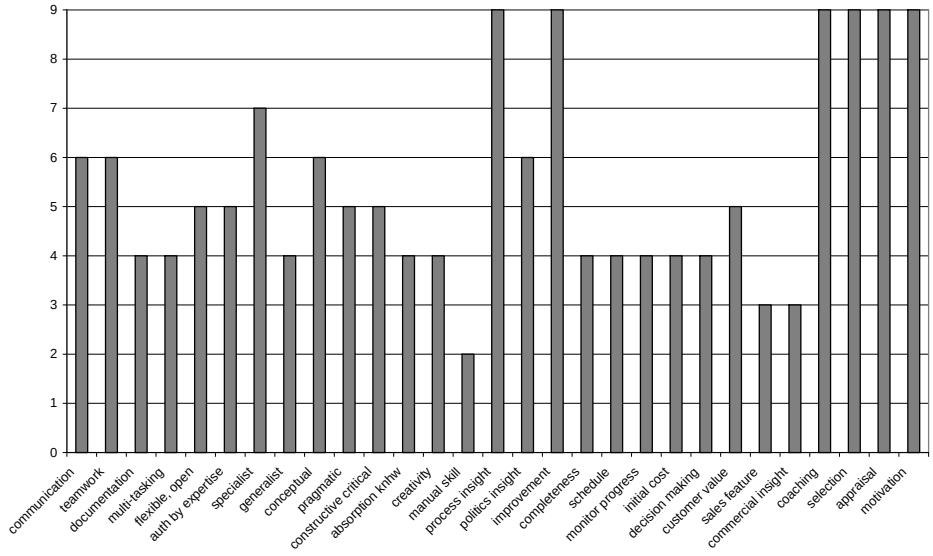


Figure 8.5: The function profile of the line manager

teamwork

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

documentation

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

multi-tasking

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

flexible, open

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

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

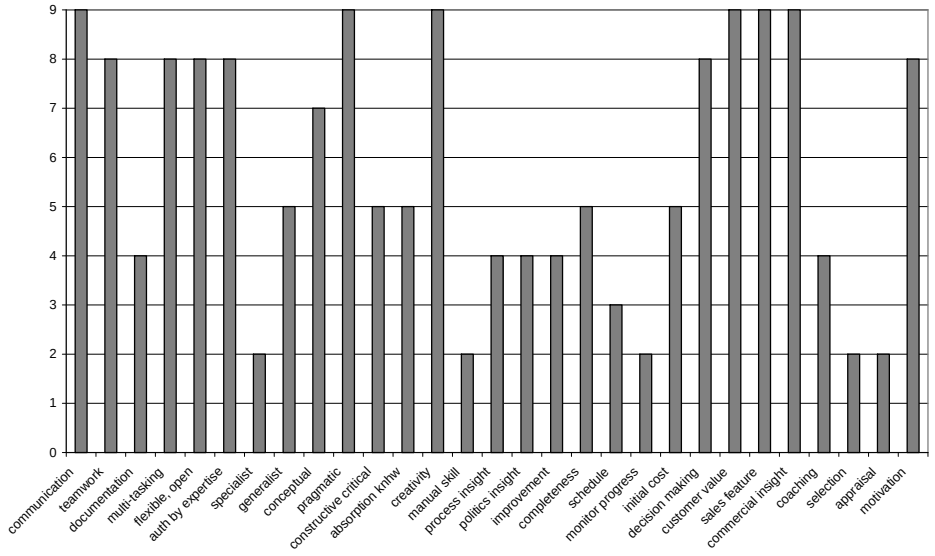


Figure 8.6: The function profile of the commercial manager

authority by expertise

The personality which convinces people by providing data, instead of citing formal responsibilities. To obtain authority by expertise hard work is required, to create a good track record and trust. Authority is earned rather than being enforced.

8.8.2 Know-how

In terms of characteristics the know-how is qualified in 2 categories, generalist and specialist, see [11].

Generalist

The persons which are always interested in the neighboring areas, how does it fit in the context?

Specialist

The persons which are always interested in better understanding, in more detail.

8.8.3 Reasoning Power

conceptual

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

pragmatic

The ability to accept non-idealities, to go after the 80% solution. The ability to connect ”fuzzy” concepts to real world implementations.

constructive critical

The ability to identify problems, formulate the problems and to trigger solutions.

fast absorption of know-how

The ability to jump into a new discipline and to absorb the required know-how in a short time.

creativity

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

8.8.4 Executing Skills

Manual Skils

The ability to **do** things, for instance build or test something. This ability is complementary to the many ”mental” skills in this list of characteristics.

8.8.5 Process Skills

process insight

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

politics insight

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

improvement drive

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

8.8.6 Project Management Skills

completeness

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

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

schedule

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

monitor progress

The ability to monitor progress and the ability to chase after the causes of delays.

initial cost

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

decision making

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

8.8.7 Commercial Skills

customer value

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

sales feature

The ability to recognize features needed to sell the product. The ability to characterize the relevant characteristics of these features (”tickmark only”, ”competitive edge”, ”show-off”, etcetera)

commercial insight

The ability to think in commercial terms and concepts, ranging from ”branding” to ”service approaches”

8.8.8 Human Resource Management Skills

coaching

The ability to coach other people; help other people by reflection, by stimulating self-deployment.

selection

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

appraisal

The ability to asses employees and to communicate this assesment in a fair and balanced way.

motivation

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

8.9 Acknowledgements

I thank Pierre America for fine tuning of translations, spelling and capitols.

Chapter 9

Roadmapping

9.1 Introduction

The definition of new products is a difficult activity, which frequently ends in a stalemate: "It must be done" vs "It is impossible to realize in such a short time frame". The rootcause of this frustrating stalemate is most often the fact that we try to solve a problem in a much too limited scope. Roadmapping is a method to prevent these discussions by lifting the discussion to a wider scope: from single product to product portfolio and from single generation of products to several years.

In addition to the scope change the roadmap is the integrating vision which shared by the main stakeholders. A shared vision generates focus for the entire organization and enables a higher degree of cooperating concurrent activities.

This article describes what a roadmap is, how to create and maintain a roadmap, the involvement of the stakeholders and gives criteria for the structure of a roadmap.

9.2 What is in a roadmap?

A roadmap is a visualization of the future (typical 5 years) integrating all relevant business aspects. Figure 9.1 shows the typical contents of a roadmap. At the right hand side the owner of the view is shown, while the left hand side shows the asymmetry of the views: the market is driving, while technology people and process are enabling.

Key to a good roadmap is the skill of showing the important, relevant issues. The roadmap should provide an immediate insight in the most relevant developments from the 5 mentioned points of view. These issues are primarily related by the time dimension.

The convention used in this article is to show products, technologies, people or process when they are or should be available. In other words the convention

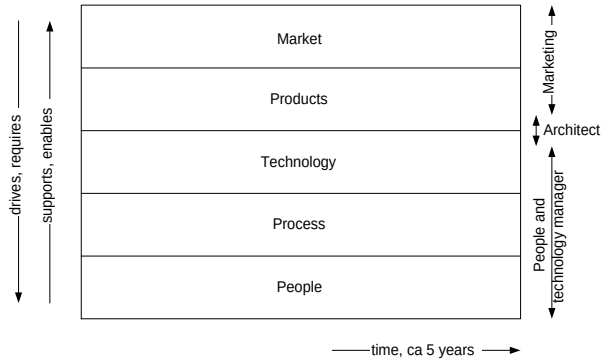
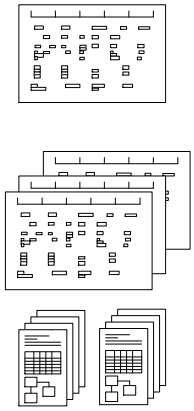


Figure 9.1: The contents of a typical roadmaps

is to be extrovert, be oriented to the outside world. The introvert question when and how to achieve these items are not directly shown, although the availability of people and process is quite often before the availability of the technology, which again predates the product or market.



Toplevel roadmap	Single page	Poster part of many presentations
Supporting roadmaps	Single page per view or per driver	Poster part of many presentations
Supporting reports	Document per relevant subject	

Figure 9.2: The roadmap is documented at several layers of detail

A good roadmap is documented and presented at several layers of detail. The higher levels are important to create and maintain the overview, while the more detailed levels explain the supporting data. Figure 9.2 shows the desired granularity of the roadmap documentation.

9.3 Why Roadmapping?

The policy and planning process [19] relies heavily on roadmapping as tool. The main function of roadmapping is to provide a shared insight and overview of the

business in time. This insight and overview enables the management of the 3 other processes:

- the customer oriented process
- the product creation process
- the people process and technology management process

Where managing these processes means defining the constraints for these processes in terms of budgets and results: Where do we spend our money and what do we get back for it?

At business units without roadmapping the following effects can be observed:

- Frequent changes in product policy
- Late start up of long lead activities, such as people recruitment and process change
- Diverging activities of teams
- Missed market opportunities

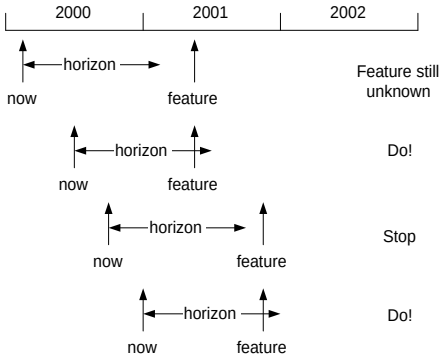


Figure 9.3: Management based on a limited horizon can result in a binary control of product policy decisions

The frequent changes in the product policy are caused by the lack of time perspective. In extreme cases the planning is done with a limited time horizon of for instance 1 year. External events which are uncertain in time can shift into the limited horizon when popular and disappear again when something else is hyping. This effect is shown in figure 9.3

The availability of a roadmap will help the operational management to perform a low pass filter on their decisions. This is shown in figure 9.4.

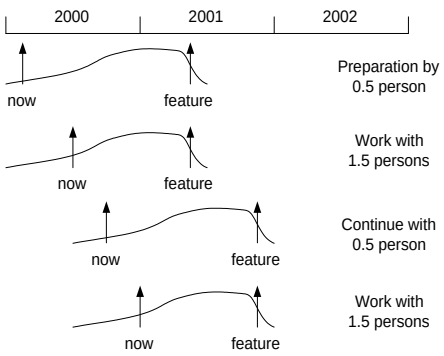


Figure 9.4: Management with a broader time and business perspective results in an analog control: work with some more or some less people on the feature

An inherent benefit of roadmapping is the anticipating value, which is especially important for everything which has a long lead time. Examples are technology, people and process. This is not limited to development, for example for manufacturing it is the same; reliable mass production has a significant lead time.

9.4 How to create and update a roadmap

A roadmap is a joint effort of all relevant stakeholders. For a typical High-tech company the stakeholders will be:

- Business manager (overall enterprise responsible)
- Marketing manager
- People and technology manager(s)
- Operational manager(s)
- Architect

An efficient way to create or update a roadmap is to work in "burst-mode": concentrate for a few days entirely on this subject. To make these days productive a good preparation is essential. Figure 9.5 shows the roadmap creation or update as three successive bursts of 2 days.

The input for first days is prepared by expert teams, which focus on the market, the product and the technology section of the roadmap. The people and process status should be available in presentable format. The target of the first burst is:

- Shared vision on market
- First iteration of possible products as an answer to the market

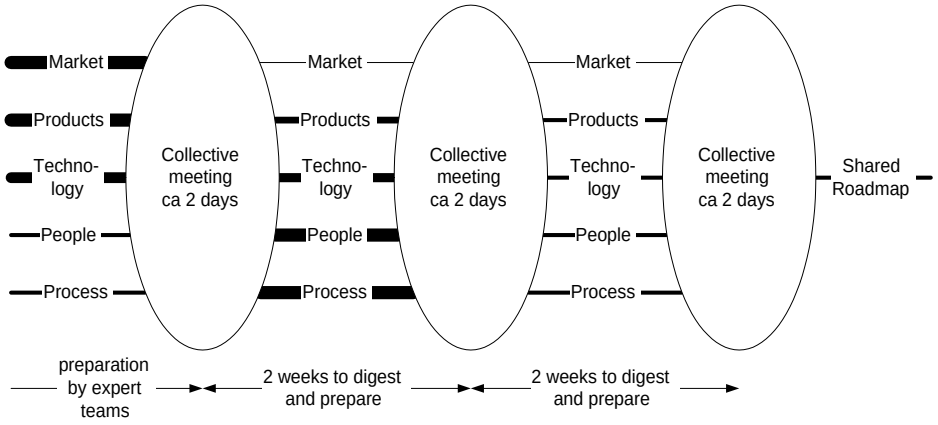


Figure 9.5: Creation or Update of a roadmap in "Burst-mode"

- Share technology status, as starting point for technology roadmap
- Explore people and technology status, to identify main issues

Between the first and second burst and between the second and third burst some time should be available, at the one hand to digest the presented material and the discussions, at the other hand to prepare the next session. The target of the second burst is:

- Obtaining a shared vision on the desired technology roadmap
- Sharing the people and process issues required for the products defined in the first iteration
- Analyzing a few scenarios for products, technologies, people, and process

The thickness of the lines in figure 9.5 indicate the amount of preparation work for that specific part of the roadmap. It clearly shows the the shift in attention from the market side in the beginning to the people and process side later. This shift in attention corresponds with the asymmetry in figure 9.1: the market is driving the business, the people and processes are enabling the business.

The function of the collective meetings is to iterate over all these aspects and to make explicit business decisions, which means that the product roadmap should be matched by the technology, people and process roadmap. Note that the marketing roadmap can deviate, in other words an explicit business decision can be made to leave market segments to the competition.

Figure 9.6 shows the roadmap activities in time. Vertical the same convention is used as in figure 9.1, the higher entities drive the lower entities in the roadmap. This figure immediately shows that although "products" are driving the technology,

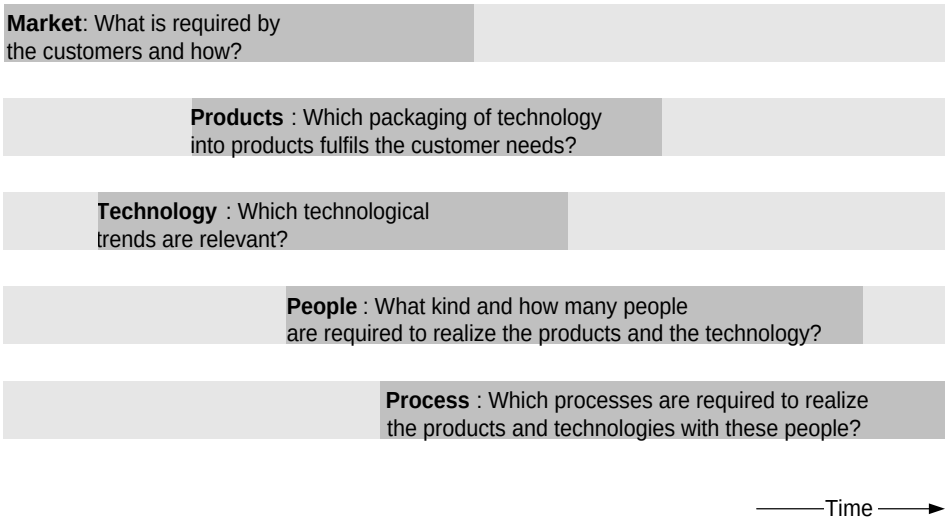


Figure 9.6: The roadmap activities visualized in time.

the sequence in making and updating the roadmap is different: the technological opportunities are discussed before detailing the product section of the roadmap.

9.5 Roadmap deployment

The roadmap is a shared vision of the organization. This vision is implemented in smaller steps, for instance by defining outputs per program and the related resource allocations per program. In Figure 9.7 it is shown that roadmap updates are performed regularly, in this figure every year. After determining the vision a "budget" is derived, which is revised with an higher update frequency, here every 3 months. The budget itself is used as the framework for the operation, which realizes the outputs defined in the budget. The operational activity itself updates the schedules again with a much higher frequency than the budget update frequency; within the operational activity the updates are mostly event driven: changes in the market, technology or resources which render the existing plan obsolete.

In other words from long term vision to short term realization is a 3-tier approach:

	horizon	update	scope	type
Roadmap	5 years	1 year	Portfolio	Vision
Budget	1 year	3 months	Program	Commitment
Detailed plan	1 mnth..1yr	1 day..1 mnth	Project or activity	Control means

The roadmap gives the context for the budget, the budget defines the context for the detailed plans.

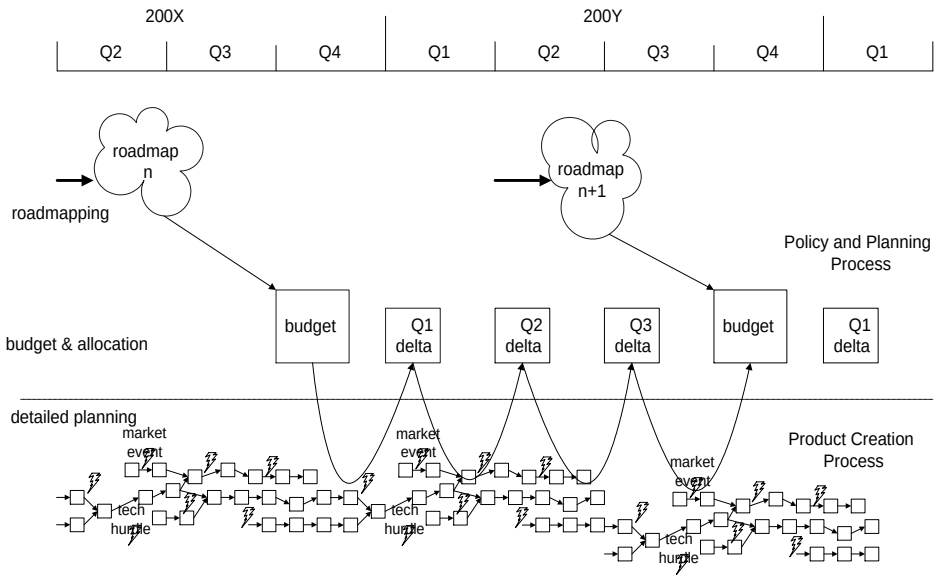


Figure 9.7: The roadmap is used to establish committed resource allocations and outputs as an baseline for development. Such a budget is updated regularly, for instance every quarter. Note that project plans change much faster, these plans are the control means for projects.

9.6 Roadmap Essentials

Each roadmap should fulfil the following requirements:

- Recognizable issues for all stakeholders
- Clear positioning in time; uncertainty can be visualized
- The main events (enabling or constraining) must be present
- Limited amount of information to maintain the overview

9.6.1 Selection of most important or relevant issues

The most essential art of making a roadmap is the selection of the most relevant issues. It is quite easy to generate an extensive roadmap with all marketing and technological events visualized, however this kind of roadmap is only the first step in making the roadmap, because the overload of information will inhibit the necessary overview.

9.6.2 Keydrivers as a means to structure the roadmap

In [15] keydrivers are explained as an effective method to capture requirements. These keydrivers can also be very helpful in the creation and update of the roadmap. At the marketing side the trend in these keydrivers must be visible in the roadmap. This also helps to structure the roadmap as well.

The supporting roadmaps should clarify how the keydriver trends will be supported. For instance a technology roadmap per keydriver is a very explicit way to visualize the relationship between the market in terms of keydrivers and technology.

9.6.3 Nothing is certain, ambiguity is normal

A roadmap is a means to share insight and understanding in a broader time and business perspective. Both dimensions are full of uncertainties and mostly outside the control of the stakeholders. It can not be repeated enough that a roadmap is **only** a vision (or dream?).

The only certainty w.r.t. a roadmap is that reality will be differ from the vision presented in the roadmap.

This means that the investment in making the roadmap more accurate and more complete should be limited. Nobody can predict the future, we will have to live with rather ambiguous visions and expectations of the future.

9.6.4 Use facts whenever possible

The previous subsection can be used as an excuse to deliver sloppy work. Unfortunately a sloppy roadmap will backfire to the author. It is recommended to base a roadmap on facts whenever possible. Sources of facts are:

- Market analysis reports (number of customers, market size, competition, trends)
- Installed base (change requests, problem reports, historical data)
- Manufacturing (statistical process control)
- Suppliers (roadmaps, historical data)
- Internal reports (technology studies, simulations)

Use of multiple data sources enable cross-verification of the sanity of assumptions. For instance predictions of the market size in units or in money should fit with the amount of potential customers and the amount of money these customers are capable (and willing) to spend.

9.6.5 Don't panic in case of impossibilities

It is quite normal that the roadmap sections appear to be totally inconsistent. For instance a frequent occurring effect is that the budget estimate in response of the market requirements is 3 times the available budget [1]. Looking back in retrospect the realized amount of work for the given budget is often twice the estimate made for the roadmap. In other words, due to a number of effects, the roadmap estimates tend to have a pessimistic bias.

The overestimation can be caused by:

- Quantization effects of small activities (the amount of time is rounded to manweeks/months/years)
- Uncertainty is translated into margins at every level (module, subsystem, system)
- Counting activities twice (e.g., in technology development and in product development)
- Quantization effects of persons/roles (full time project leader, architect, product manager, etcetera per product)
- Lack of pragmatism (technical ambition is not too bad during the roadmap process, as long as it does not pre-empt a healthy decision)
- Too many bells and whistles without business or customer value

9.7 Roadmap example

Due to the strategic value of roadmaps for the business it is impossible to illustrate the article with real actual roadmaps. Figure 9.8 shows an academic example of a highest level roadmap.

This example shows that the highest level roadmap should fit on a single A4. Presented on an overhead sheet or projected by means of a beamer the text should be readable for the audience. As can be seen from this example this requirement is quite a challenge. For the overview it is essential to show the information entirely, to enable everyone to see the broader perspective and to see the many underlying relationships.

Supporting roadmaps can concentrate on specific relations, for instance between keydriver and the required technology. These supporting roadmaps should be

[1] This factor 3 is an empiric number, which of course depends on the company and its culture. In one of the companies I have worked for the pragmatic Anglo-Saxon culture had a somewhat smaller gap between the estimated requirement and the available budget. In this company a well supported shortage was taken seriously by the management, which in return might have resulted in less defensive overestimations.

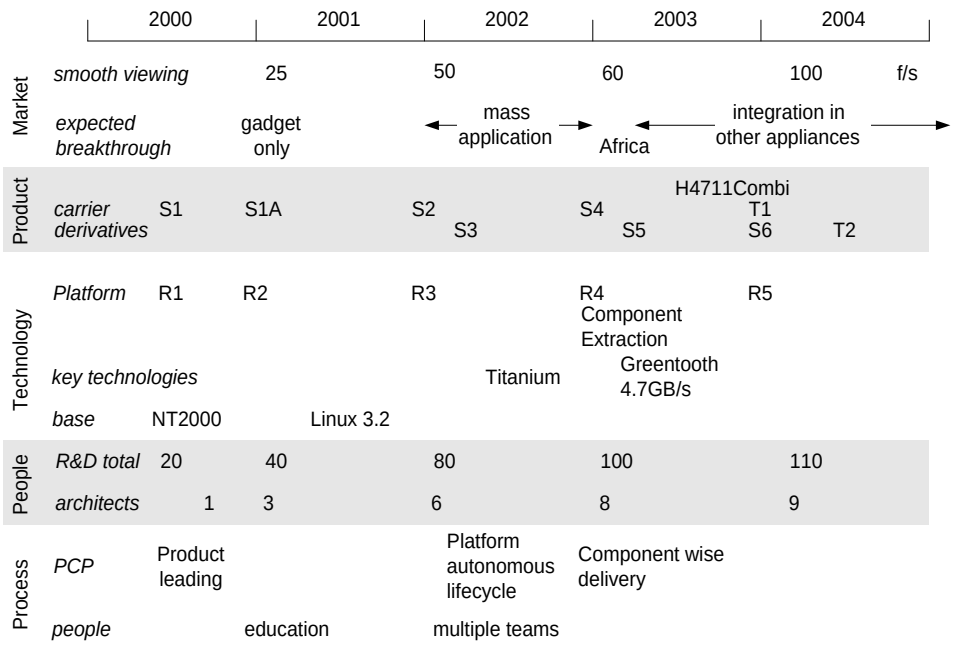


Figure 9.8: An academic example of a roadmap, to illustrate the structure of the highest part of the roadmap

linked to the highest level roadmap by the time axis and a small set of recognizable landmarks, for instance quantified keydrivers and the main products.

9.7.1 Time Axis

Every roadmap has a time axis, where the left hand side should correspond to the near future (between the day of creation and half a year in the future). The right hand side typical is 5 years in the future. For supporting roadmaps both zooming in and zooming out can help to bring a specific message. For instance for a subsection of technology it might be useful to show the next 3 years only, if a lot of information is present in that period. To make some marketing or technology trends clear a larger span of time might be useful, for instance for Moore's law a 10 or even 15 year window might be required.

In some cases it is also helpful to show the historical context. Historical data is an important source of information. For instance in the lithography market it is relevant to know the introduction dates of new exposure wavelength. A single introduction date is not enough in the lithography case, the introduction lead times play an important role in the product definitions. Hence showing the R&D use and showing the volume production use for historical wavelength transitions helps to get an historical perspective.

9.7.2 Vertical axis

The vertical axis describes the subject in the roadmap. The main division in market, product, technology, people and process is visually supported by the background color. The headers are required as legend for the less experienced roadmap readers, however these should use as little space as possible, from information point of view this is overhead. Quite often more vertical structure is present, for instance grouping towards market, product or technology type, grouping per keydriver etcetera. In figure 9.8 examples are *smooth viewing* as keydriver, *expected market breakthroughs* and *Platforms*. Again this structuring can be made explicit by showing it at the vertical axis, where it should look different from the roadmap contents itself. Here also the space usage should be minimized.

Space in the roadmap diagrams is the most scarce resource. Common information is sometimes present only once, for instance the units for the frame rate *f/s* is at the right hand side of the diagram. The information in the center of the roadmap is reduced to the essence, although it should stay recognizable.

9.7.3 Market

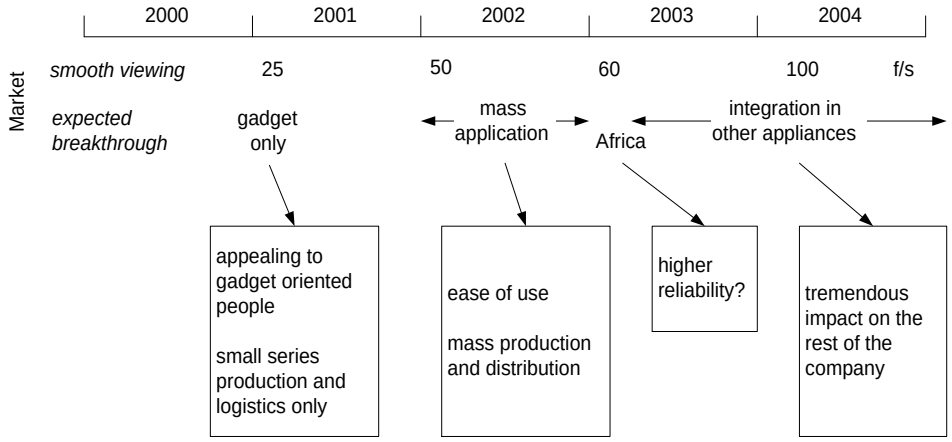


Figure 9.9: The market part of the highest level roadmap

In this example only 1 keydriver *smooth viewing* is shown. This keydriver is here expressed in terms of a frame rate, which in the user community appears to be the dominating parameter to express the smoothness of viewing. The numbers here indicate the user expectations with respect to this frame rate. This expectation is the result of the general perception of technology and the competition at the one hand and the real need at the other hand. It might well be that at a certain frame rate, the focus shifts towards resolution. If that is the case this should be visible in the roadmap.

Although shown in the market part are market relevant breakthroughs which are expected. In this example it is expected that the first generation of products will be *gadget only* products. This single phrase will result in a cascade of requirements and decisions, ranging from *appealing to gadget oriented people* to *small series production and logistics only.*

The next breakthrough in this example is the use of the product in the mass market. An uncertainty of one year is shown for this breakthrough, the opinion is that it will happen, however the timing is somewhat uncertain. Again this single phrase hides a world of requirements such as *ease of use* and *mass production and distribution.*

Once the product is well established it is expected to become integrated in different appliances. The timing of this breakthrough is inherently more uncertain. The impact of this market trend for the company and the rest of the roadmap is tremendous a well defined self-sustained product suddenly becomes an integrated part in some other product.

The last breakthrough shown here is the expectation that the product will be introduced in Africa as well. This might for instance result in higher reliability requirements.

9.7.4 Products

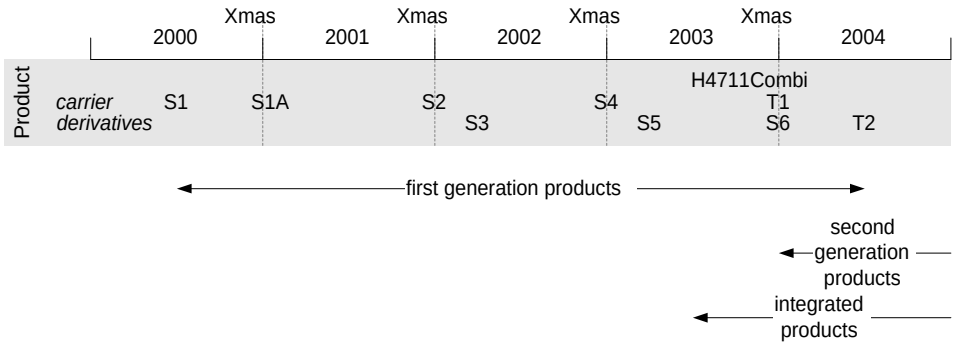


Figure 9.10: The product part of the highest level roadmap

A number of products are synchronized with the market heartbeat, for instance the christmas buying period. In the example the products S1A, S2, S4, S6 and T1 are explicitly synchronized with this external event. A number of derived products or improved products will be introduced at a different moment. The products on the *carrier* line serve as a carrier for the platform technology development, this shows a relationship between the product section and the technology section.

The product identification gives some indication of the expected product content. The numbering suggest some improvement/extension of the same type of product.

Going form S1 to S1A suggests an improvement only. The T1 appears to be a new product of a new generation of the same product which co-exists for some time with the old product line (S6).

A very challenging product in this part of the roadmap is the H4711combi, which is an existing H4711 product with the functionality of the S5 product integrated. This product is in fact a reference to the next level of roadmapping at portfolio level.

9.7.5 Technology

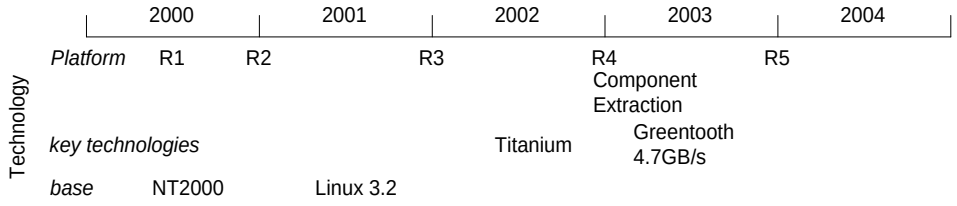


Figure 9.11: The technology part of the highest level roadmap

The technology roadmap shows the expected timing of the platform developments. In this example the platform development is coupled to a carrier product: Release 1 is coupled to product S1, release 2 to product S1A and so on until release 5, which is coupled to product T1. The platform technology becomes available at the same time as the corresponding product.

The next line shows that a component like approach will become available at the same time as the release 4 platform. This component approach is needed for the H4711combi product, which is derived from an totally different product.

The line *key technologies* shows the expected timing of 2 technology breakthroughs in the outside world, which will have an high impact on the products. The *Titanium* Gigaprocessor will enable the flexible component based architecture, while the *Greentooth* communication infrastructure will remove data transfer bottlenecks, which is important to realize the required frame rate.

The *base technologies* line show the relevant events in the base technology, in this case the operating system which is used. Apparently a switch is foreseen from NT2000, which is rather monolithic and therefore memory intensive and expensive NT2000, to the configurable Linux variant. The version of Linux to be used is the stable version (.2) of the next generation of Linux (3).

9.7.6 People

The people roadmap in this example is kept rather limited. The total need of developers (*R&D total*) is shown at the beginning of every year. The discussions about headcount are synchronized with the calendar year, the budget discussions take

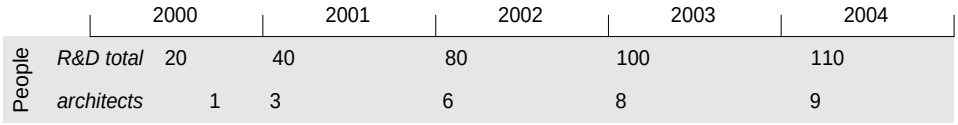


Figure 9.12: The people part of the highest level roadmap

place at the end of the year and discuss the headcount in terms of january 1. By using the same convention in the roadmap the essential numbers are comparable.

Next in this section of the roadmap the most critical resource is shown. In view of the fact that this article fits in the Gaudí articles, which describe system architecture, it is clear that architects are seen as the most critical resource[2].

The roadmap at this level does not yet spell out the problem behind these number, which is that the lead time to acquire or educate architects is quite long. Acquisition of new architects will cost between half and one year, followed by a period from 1 to 3 years to become productive and to operate at architect level. Education of R&D people already present also costs between 1 and 3 years. Unfortunately only 10% of a typical R&D population has the potential skills to become an architect. In this example about 2 existing developers will be capable to grow into this role, hence the other architects will have to be recruited.

9.7.7 Process

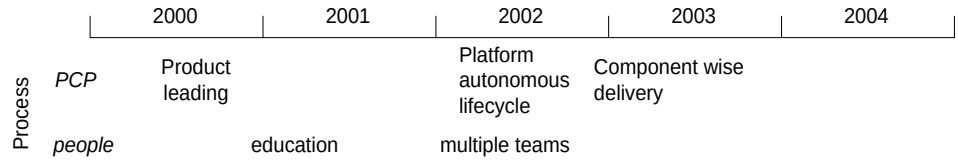


Figure 9.13: The process part of the highest level roadmap

The core *PCP* process in the beginning is product oriented. The technology consolidation in a platform is a spinoff from the product development. After the 3^{rd} iteration of the platform the product development and the platform development are decoupled. This allows for lifecycle independent development of the platform. Tight coupling of product and platform development is a complicating factor when many products are derived from the platform.

Forced by the vision that integrated products are required a further decoupling is required. In the technology section this decoupling is shown as *Component*

[2] Although it is partly a joke, reality is that in many cases the architects are truly the most critical resource.

extraction. A critical success factor for working with a component strategy is that the processes need to be in place to create, change, manage and deploy components. These processes are summarized in the highest level roadmap as*component wise delivery.*

The strong increase in the amount of developers will also have a dramatic impact on the processes in use. When the group reaches a size of approximately 80 people, it should be working in multiple relatively autonomous teams. This requires that processes are in place to split up the work, but also to manage the overall consistency and balance. The phrase *multiple teams* is used to reflect this requirement.

9.8 Bootstrapping the roadmapping process

Many companies and business units have no ongoing roadmapping activity or only a limited roadmapping activity, for instance a product roadmap only. It is a daunting task for a system architect to introduce a roadmapping process as described until now.

Introduction of a roadmapping process must be viewed as part of a change management process. Successful introduction of roadmapping coincides with changes in all aspects of the business.

Important heuristics of change management are:

- People don't want to **be** changed. They are quite often willing to change.
- Changing the way of working or the culture costs many years.
- Work at multiple tracks at the same time, a.o. managerial, operational, strategic, etc.
- Earn credit by showing usable results.

Based on these heuristics it is clear that the introduction of roadmapping should be done in a number of smaller steps. The motto here is: *Think big, act small.*

Figure 9.14 shows the bootstrapping of the roadmap process, which typical will take 2 to 4 years. The benefits of starting the process are available more or less immediately, the ultimate maturity with the related efficiency costs the mentioned 2 to 4 years.

A good start is to capture the existing visions, plans, budgets, etcetera and integrate this information into a -1 order roadmap. In most cases this forces the stakeholders to reflect on the current status, which in most cases is rather unbalanced (for instance the first half year is covered in minute detail, the latter period is fuzzy) or it appears to be totally inconsistent (for instance marketing has an

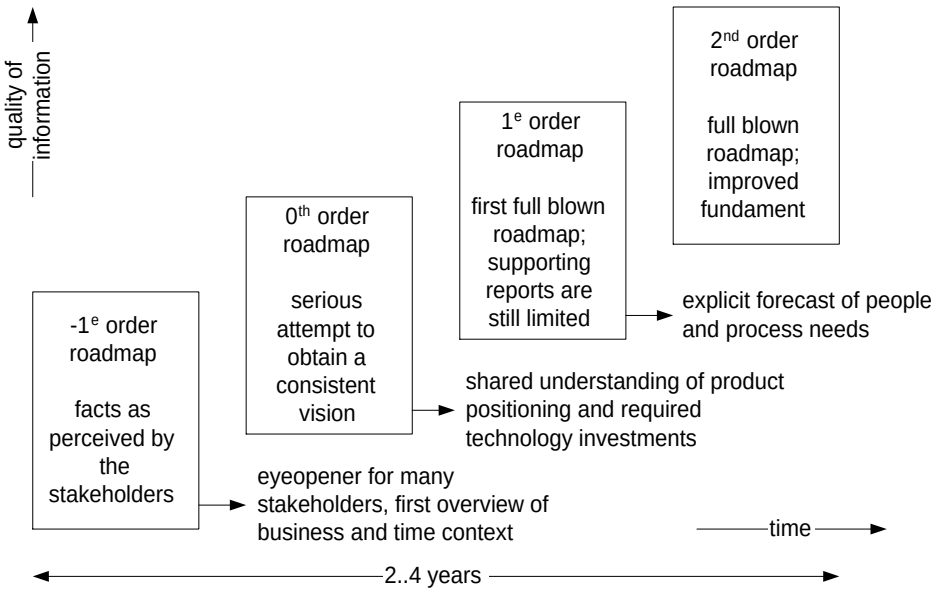


Figure 9.14: Bootstrapping the Roadmap Process

entirely different expectation than development). Best case people suddenly get an overview and gain insight in the broader context.

The result of the -1 order roadmap is that the architect gains credit and that the stakeholders are motivated to change a little bit and are willing to make a next step, for instance to make a 0 order roadmap.

A 0 order roadmap is the first attempt to get the market, the product and the technology roadmap in place. Such a partial roadmap again helps to earn credit, but it also helps to keep the stakeholders involved. Critical aspect here is the team building aspect. Roadmapping is a team activity, which requires mutual respect and trust, to enable the open and critical communication needed for the selection of the truly essential issues in the roadmap.

The entire roadmapping process is a repetition of the same activities, visualized in figure 9.15. Of course the 4 steps are not entirely sequential, they represent the main flow of the process.

9.9 Acknowledgements

The insight that a roadmap should cover all 5 views form market to process came to me via Hans Brouwhuis. Roadmapping as a business tool gained momentum within Philips during the quality actions inspired by Jan Timmer.

The critical and constructive remarks by Jürgen Müller helped to shape this article.

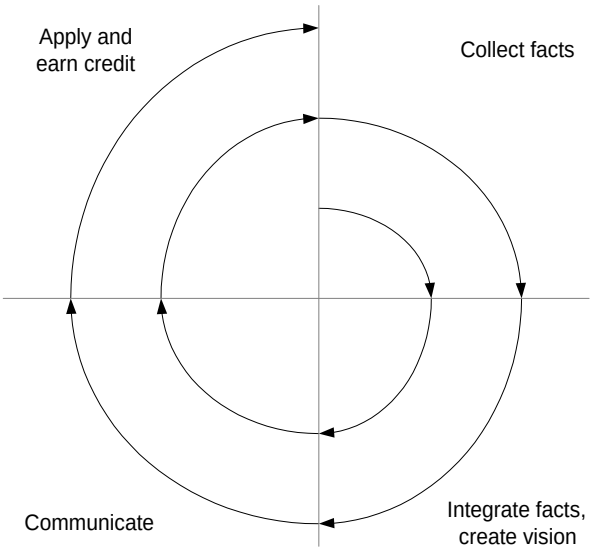


Figure 9.15: Bootstrapping the roadmap process requires a repetition of 4 steps, as visualized by this spiral

Chapter 10

Requirements Capturing by the System Architect

10.1 Introduction

The basis of a good system architecture is the availability and understanding of the requirements. This article describes how a system architect can capture the requirements and how to use these requirements in the context of the product creation process. This article builds upon the architecture process positioning as described in [20].

This article is part of the deliverables of the Gaudí project [17], which will describe other processes and methods mentioned in this article, like roadmapping.

10.2 Definition of Requirements

The term requirement is quite heavily overloaded in Product Creation context. One major interpretation is :

The requirements describe the needs of the customer.

In this article this meaning will be captured in the term *Customer Requirements.*

Another major interpretation is:

The requirements describe the characteristics of the final resulting product.

This article will use the phrase *Product Specification* when this interpretation is intended.

In the system engineering world the term *Requirement Management* or *Requirement Engineering* is being used. This term goes much farther than the two previous interpretations. The requirement management process or requirement engineering process deals with the propagation of the requirements in the product specification towards the requirements of the subsystems defined by the first design decomposition finally towards the requirements of the atomic components. In fact the

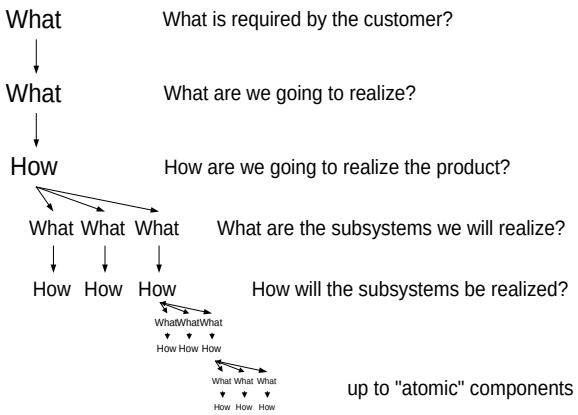


Figure 10.1: The flow of requirements

definition of the Product Specification is recursively applied for every decomposition level. On top of that the management process manages the relationships of the different aggregation levels.

Figure 10.1 show the flow of the requirements starting at the customer level.

A consensus seems to be present about the fact that requirements deal with the *what* and do not describe the *how*.

Besides the customer an important source of requirements is the producing company itself, the needs of the company itself are described in this article as *Operational Requirements.*

This article will address all the interpretations of requirements given above.

10.3 Stakeholders

A simplified process model is shown in figure 10.2. The stakeholders of the requirements are of course the customers, but also a number of representatives in the customer oriented process and most people active in the Product Creation Process.

For convenience the word customer is used, although the customer can be a business or even a group of businesses. A good understanding of the customer business is required in order to identify the customer-stakeholders.

10.4 Requirements for Requirements

Standards like ISO 9000 or methods like CMM prescribe the requirements for the requirement management process. These requirements are:

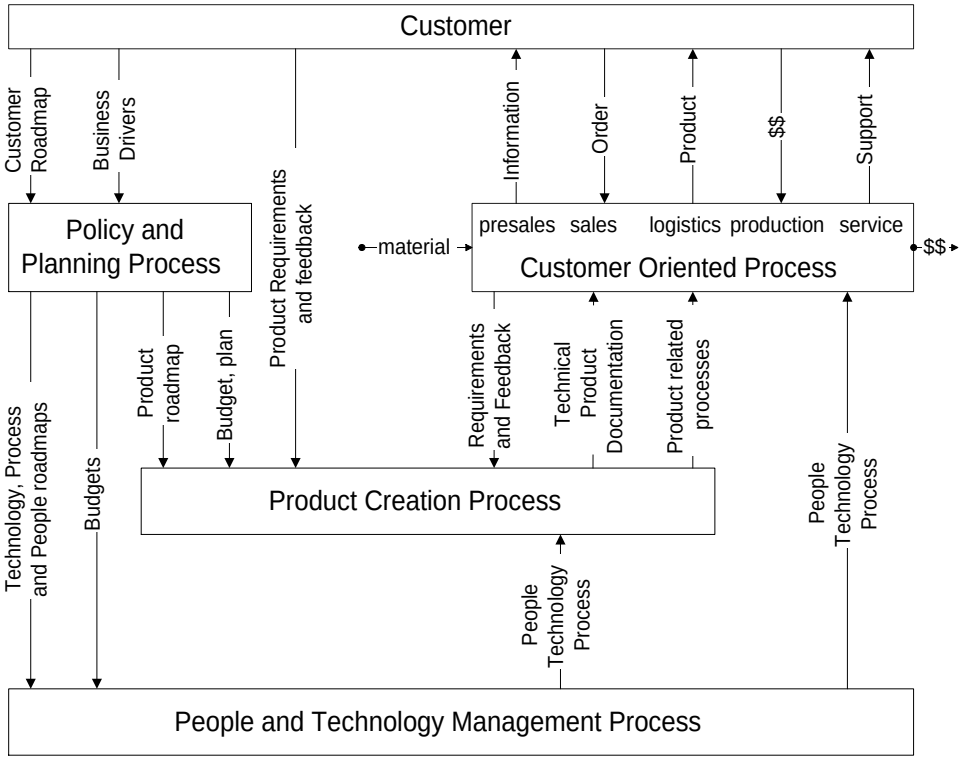


Figure 10.2: A simplified process decomposition of the business. The stakeholders of the requirements are beside the customer self, mainly active in the customer oriented process and the product creation process.

- Specific (1)
- Unambiguous (2)
- Verifiable (3)
- Quantifiable (4)
- Measurable (5)
- Complete (6)
- Traceable (7)

Unfortunately these requirements are always biased towards the formal side. A process which fulfil these requirements is from technical point of view sound and robust. However an important aspect which is forgotten quite often is that product creation is a human activity, with all their human capabilities and constraints. The

Human point of view adds a number of requirements, which are required for **every** stakeholder:

- Accessible (8)
- Understandable (9)
- Low threshold (10)

These requirements, which are imposed by the human element, can be conflicting with the requirements which are prescribed by the management process. Many problems can be traced back to violation of the human imposed requirements. For instance a customer requirement which is described so abstract that no real customer can understand it anymore is a severe risk, because early validation is impossible.

10.5 Viewpoints on Requirements

Many complementary viewpoints are required to collect the requirements. Figure 10.3 shows a useful number of viewpoints when collecting requirements.

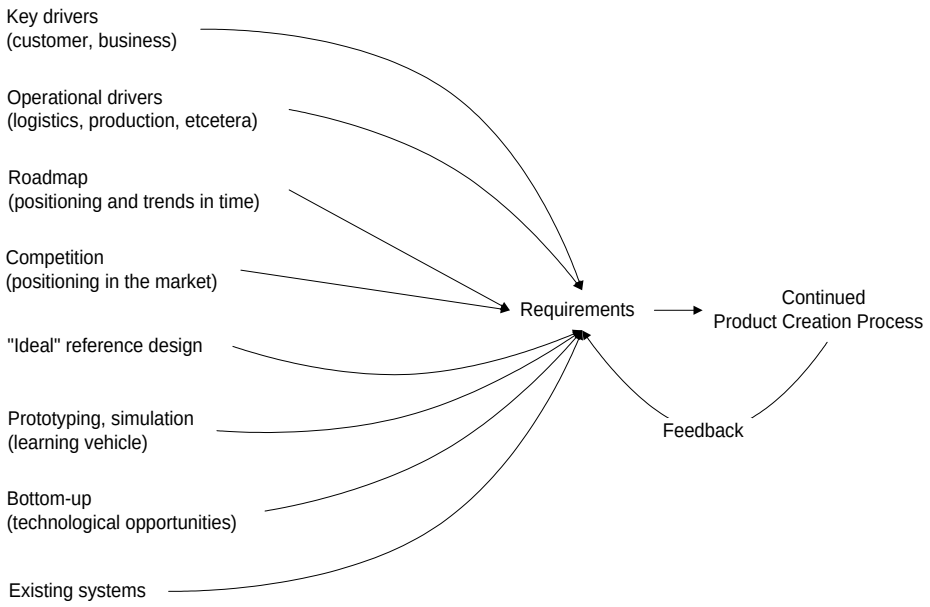


Figure 10.3: Complementary Viewpoints to collect requirements

The **keydriver** viewpoint and the **operational** viewpoint are the viewpoints of the stakeholders which are "consuming" or "using" the output of the product creation process. These viewpoints represent the "demanding side".

The **roadmap** and the **competition** viewpoint are viewpoints to position the requirements in time and in the market. Those viewpoints are important because they emphasize the fact that a product is never made in isolation, but in a rather dynamic and evolving world.

The **"ideal" reference design** is the challenge for the architect. What is in his vision the perfect solution? From this perfect solution the implicit requirements can be reconstructed and added to the rest of the requirements.

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, which do not appear with a pure paper approach.

The **bottom up** viewpoint is the viewpoint which takes the technology as the starting point. This viewpoint sometimes triggers new opportunities which are overlooked by the other viewpoints due to an implicit bias by today's technology.

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

The requirement 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 requirements as well. In fact nearly all viewpoints are present and relevant during the entire Product Creation Process.

10.6 Reference Architecture and Key Drivers

A system architect must look at the product from multiple complementary viewpoints. Figure 10.4 shows 5 useful views for a reference architecture.

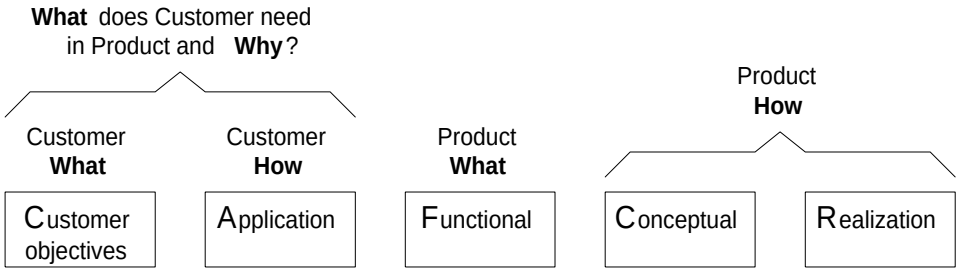


Figure 10.4: A Reference Architecture views the architecture from 5 viewpoints

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.

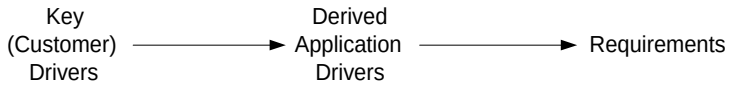


Figure 10.5: The mapping of Key Drivers via derived application drivers on requirements

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 maintain focus on the essence, the name is on purpose **Key** driver. The key drivers are one aspect of the business architecture. Table 10.1 shows some recommendations with respect to the definition of key-drivers.

- Limit the number of key drivers, minimal 3, maximal 6.
- Make the scope specific in terms of customer or market segments.
- Start the definition process with facts, for instance by asking **why** questions about the functionality and performance of existing products.
- Don't leave out the obvious key drivers, for instance the well-known main function of the product.
- Choose short names, recognized by the customer.
- Choose 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".)

Table 10.1: *recommendations with respect to the definition of 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.

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.

10.7 Example Motorway Management

Figure 10.6 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.

10.8 Requirements Value and Selection

The set of customer requirements and operational requirements 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 requirements as input. Figure 10.7

The selection process is primarily controlled by the strategy of the company, which determines market, geography, timing and investments. The roadmap, which is in itself based on the strategy, is giving context to the selection process for an individual product. The reality of the competitive market is the last influencing factor on the selection.

The selection will be based on facts and estimates from the technology, people and process world, which will often constrain the possibilities.

The amount of requirements sometimes asks for a first selection step, which determine the "obvious". For some requirements it is immediately obvious that

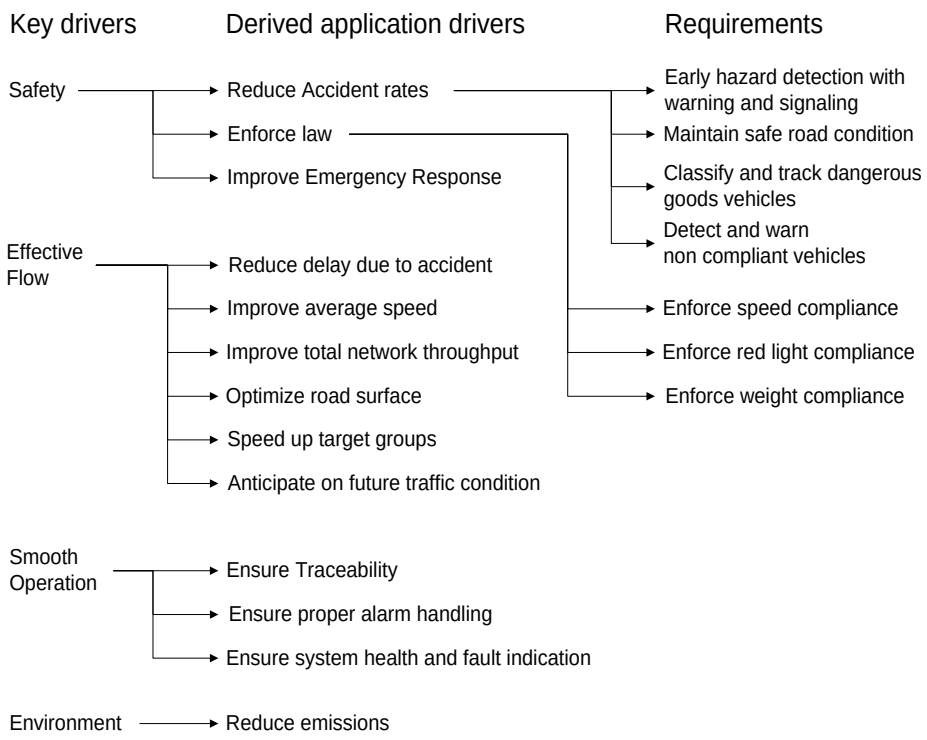


Figure 10.6: The key drivers, derived application drivers and requirements of a Motorway Management System

they have to be done anyway, while other requirements can be delayed without any problem. Figure 10.8 shows a number of qualitative characterizations of requirements, visualized in a two-dimensional matrix. For every quadrant in the matrix a conclusion is given, a requirement must be done, not be done or must be discussed further.

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

- importance versus urgency
- customer value versus effort
- must have

In the final selection step a more detailed analysis step is preferable, because this improves the understanding of the requirements 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, for instance:

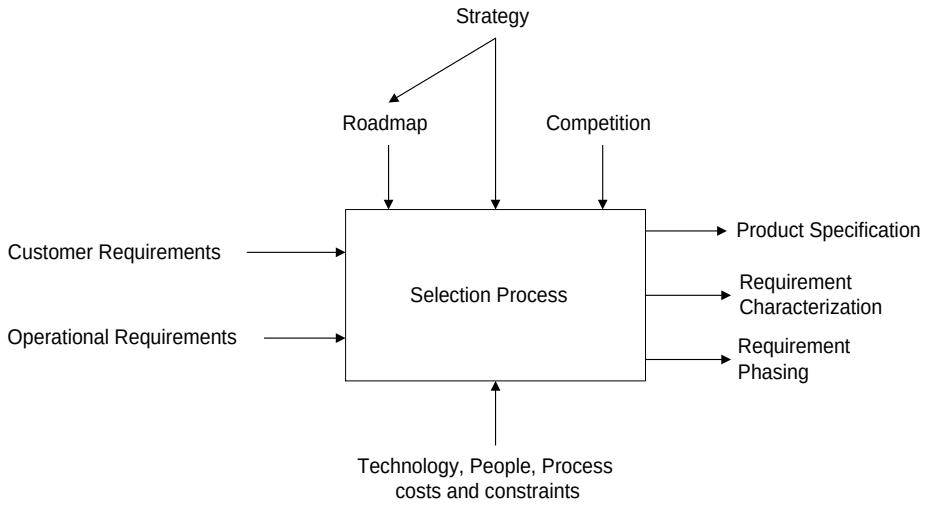


Figure 10.7: The selection process produces a product specification and to prevent repetition of discussion a phasing and characterization of requirements

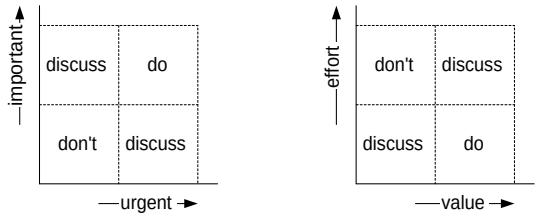


Figure 10.8: Simple methods for a first selection

- Value 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

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

The output of the requirement characterization and the proposed phasing is

input for the next update cycle of the roadmap. Note that some companies use the word roadmap for the phasing of the requirements, while this article uses a roadmap in a much broader sense, see the article [16]

10.9 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.

Chapter 11

Product Families and Generic Aspects

11.1 Why generic developments?

Many people advocate generic developments, claiming a wide range of advantages, such as listed in table 11.1.

Effective implementation of generic development has proven to be quite difficult. Many attempts to achieve these claims by generic developments have resulted in the opposite goals, such as increased time to market, quality and reliability problems etcetera. We need a better rationale to do generic developments, in order to design an effective generic something creation process.

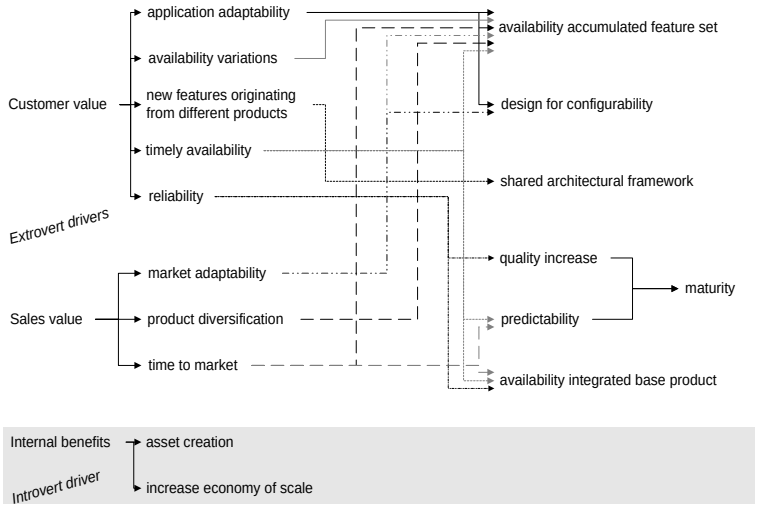


Figure 11.1: Drivers of Generic Developments

- Reduced time to market (11)
- Reduced cost per function (12)
- Improved quality (13)
- Improved reliability (14)
- Easier diversity management (15)
- Increases uniformity (16)
- Employees only have to understand one base system (17)
- Improved predictability (18)
- Larger purchasing power (19)
- Means to consolidate knowledge (20)
- Increase added value (21)
- Enables parallel developments of multiple products (22)
- “Free” feature propagation (23)

Table 11.1: *Advantages which are often claimed for generic developments*

Figure 11.1 shows drivers for Generic Developments and the derived requirements for the Generic Something Creation Process. The first 2 drivers (*Customer value* and *Sales value*) are extrovert: does the product have value for the customer and is he willing to buy the product? The last driver *Internal Benefits* is introvert, it is the normal economic constraint for a company.

Today high tech companies are knowhow and skill constrained, in a market which is extremely fast changing and which is rather turbulent. Cost considerations are degraded to an economic constraint, which is orders of magnitude less important than being capable to have valuable and sellable products.

The derivation of the requirements shows clearly that these requirements are not a goal in itself. For instance an shared architecture framework is required to enable features developed for one product to be used in other products as well, which in turn should have value for a customer. So the verification of this requirement is to propagate a new valuable feature from one product to the next, with small effort and lead time.

These drivers and requirements derivation is emphasized, because many generic developments result in large monolithic general purpose things, fulfilling:

- availability accumulated feature set
- designed for configurability
- shared architectural framework
- mature

without bringing any of the customer or sales value; "You can not have this easy shortcut, because our architectural framework does not support it, changing the framework will cost us 100 man-years in 3 years elapsed time"

11.2 Granularity Of Generic Developments

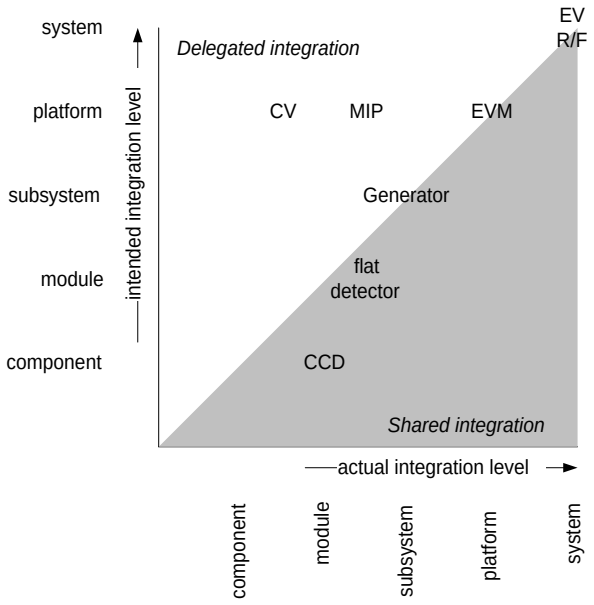


Figure 11.2: Granularity of generic developments shown in 2 dimensions.

Figure 11.2 shows the granularity of generic developments in 2 dimensions. The vertical dimension is the preparation level: What is the intended scope of the generic developments, how far is the deployment prepared? The horizontal dimension is the integration level: How far are the generic developments integrated when the "product developers" deploy the generic development?

Both axis range from (atomic) component until (configurable) system. Developments on the diagonal axis, which have a scope where the preparation level is equal to the integration level, are straightforward developments in which the

integration takes place as far as autonomously possible. Some generic developments concentrate on the generation of building blocks, leaving ("delegating") the integration to the product developer. For rather critical generic developments the generic development surpasses its own deliverable to ensure the correct performance of the generic something in its future context(s).

In these figures a number of medical generic developments are shown, as an example for the categorization.

An extreme example of "delegated" integration is CV, which stands for Common Viewing. This is an attempt to benefit from generic developments at the end of the eighties. The vision was to create a large "toolbox" with building blocks which could be used to derive a wide variety of medical products ranging from MR scanners to X-ray systems. Based on OO techniques and supporting a very high degree of configurability a powerful set of (mostly SW) components was created.

The CV toolbox proved difficult to sell to product developers, amongst others due to the low integration level. The perception of the product developers was that they still had to do the majority of difficult work: the integration. The vision of a marketing manager changed the direction of CV into a completely integrated product: EasyVision R/F (EV R/F). This medical workstation for the URF (Universal Radiography Fluoroscopy) market was highly successful as an intelligent print server. The communication and print function were highly configurable to make the product adaptable to its environment.

The EasyVision R/F was used as a basis for a whole series of medical workstations and servers. The shared functionality is developed in a generic development at platform level. This platform is nowadays called EVM (EasyVision Modules). Despite its name it has still a significant integration level, with its upside (product developers are not bothered with the lower level integration) and its downside (predefined functionality and behavior).

The old CV vision is revived and a second generation of EVM is being created, covering the EVM platform functionality with a module level of integration.

11.3 Modified Process Decomposition

In [13] a simplified process description is given for a company. This decomposition assumes that product creation processes for multiple products are more or less independent. When generic developments are factored out for strategic reasons an additional process is required to visualize this. Figure 11.3 shows the modified process decomposition (still simplified of course) including this additional process "Generic Something Creation Process".

Figure 11.4 shows these processes from the financial point of view. From financial point of view the purpose of this additional process is the generation of strategic assets. These assets are used by the product generation process to enable

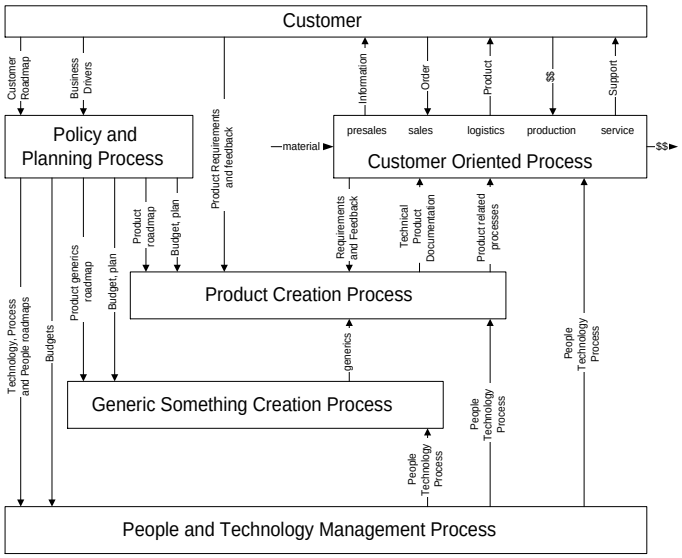


Figure 11.3: Modified process decomposition

tomorrow's cashflow.

The consequence of this additional process is an lengthening of the value chain and consequently a longer feedback chain as well. This is shown in figure 11.5. The increased length of the feedback chain is a significant threat for generic developments.

11.4 Modified Operational Organization PCP

The operational organization of the Product Creation Process is described in [13]. This organization is a straightforward hierarchy, where the limited amount of relations (conflicts) between products or subsystems are managed at the closest hierarchical management level.

Introduction of generic developments complicates the operational structure significantly[1]. Figure 11.6 shows the operational organization of the Product Creation Process, with the necessary additions to support generic developments.

The conventional Product Creation Process is based on a relative straightforward hierarchy, where the control flow and delivery flow are opposite, but map directly on the hierarchy. The introduction of generic developments breaks this simplicity: a generic development delivers to multiple product developments, while

[1]The complication can be avoided by working sequentially. However in today's dynamic market this results in unacceptable lead times. Concurrent development is a fact of life, any further reduction of lead times is welcomed!

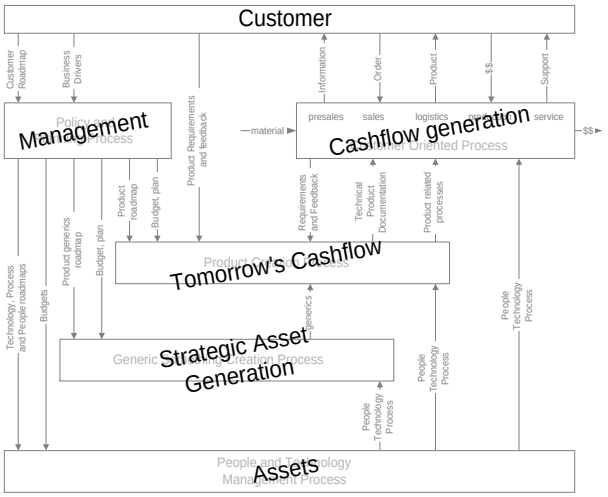


Figure 11.4: Financial viewpoint of processes

the control is taking place from an encompassing operational level, to enable operational balancing of products and generic developments. In other words the principal has is now one else than the customer.

Every operational entity needs the 3 complementing processes in the product creation process: operational management, design control and commercial. For each of these processes a role is required of someone responsible for that process: the operational manager, the architect and the commercial manager. Together these 3 people form the core team of the operation. Introduction of generic developments also requires the introduction of these roles for platform and or components.

For the architect role this means that a platform architect is needed, who is closely working together with the platform project leader and the platform manager. At the other hand he needs many architectural contacts with the product family architect, acting as the architectural principal, with the product architect, acting as customers and with the component architects, acting as suppliers.

In [7] 3 operational entities with related processes and roles are identified, see table 11.2.

abbreviation	description	maps to operational entity
AFE	Application Family Engineering	Product Family
CSE	Component System Engineering	Component
ASE	Application System Engineering	Product
-	-	Platform

Table 11.2: *The Griss Re-use Model*

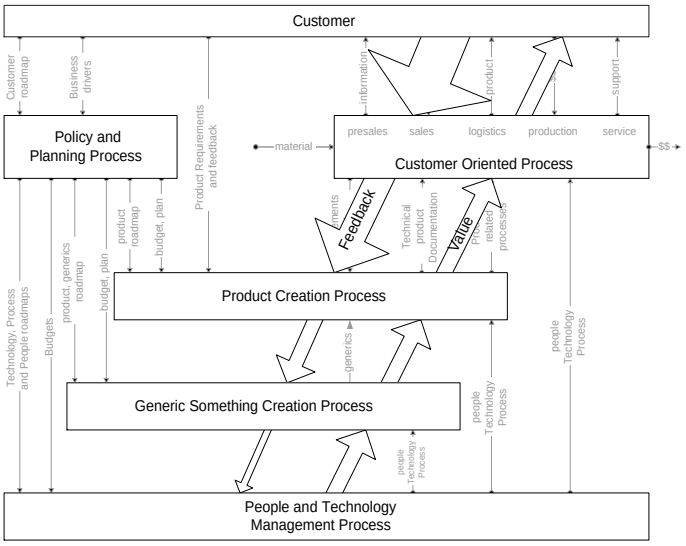


Figure 11.5: Feedback and Value flow

The operational entities from figure 11.6 are shown in the last column, although other mappings are possible too. Any mapping has the problem that 4 operational entities are represented in only 3 processes in [7]. In practice the result is that one of the roles is missing, or played implicit. For instance quite often the application family engineer starts to play platform architect, forgetting his original task *application family* engineering.

11.5 Models for Generic Developments

Many different models for the development of generic things are in use. An important differentiating characteristic is the driving force, which often directly relates to the de facto organization structure. The main flavors of driving forces are listed in table 11.3.

- Lead customer
- Carrier product
- Platform
- Technology push

Table 11.3: *List of driving forces for development*

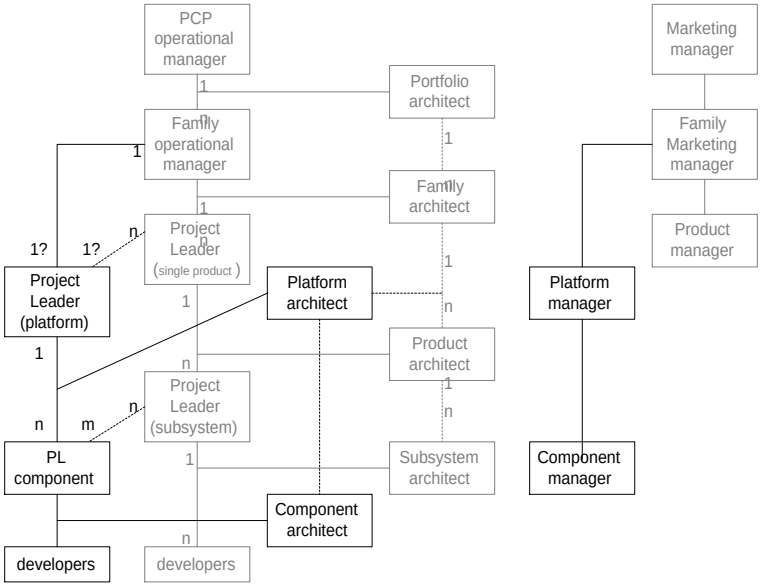


Figure 11.6: Operational Organization of the Product Creation Process, modified to enable generic developments

11.5.1 Lead Customer

The lead customer as driving force guarantees a direct feedback path from an actual customer. Due to the importance of feedback this is a very significant advantage. The main disadvantages of this approach are that the outcome of such a development often needs a lot of work to make it reusable as a generic product. The focus is on the functionality and performance, while many of the quality aspects are secondary in the beginning. Also the requirements of this lead customer can be rather customer specific, with a low value for other customer.

11.5.2 Carrier Product

The combination of a generic development with one of the product developments also shortens the feedback cycle, although it is not as direct as with the lead customer. Combination with a normal product development will result in a better balance between performance and functionality focus and quality aspects. Disadvantage can be that the operational team takes full ownership for the product (which is good!), while giving the generic development second priority, which from family point of view is unwanted.

In larger product families the different charters of the product teams creates a political tension. Especially in immature or power oriented cultures this can lead to horrible counterproductive political games.

Lead customer driven product development, where the product is at the same time the carrier for the platform combines the benefits of the lead customer and the carrier product approach. In my experience this is the most effective approach of generic developments. A prerequisite for success is an open and result driven culture to preempt any political game mentioned before.

11.5.3 Platform

In maturing product families the generic developments are often decoupled from the product developments. In products where integration plays a major role (which are nearly all products) the generic developments are pre-integrated into a platform or base product, which is released to be used by the product developments.

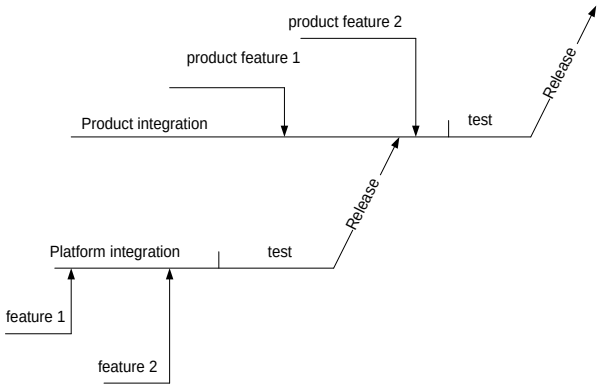


Figure 11.7: The introduction of a new feature as part of a platform causes an additional latency in the introduction to the market.

The benefit of this approach is separation of concerns and decoupling of products and platforms in smaller manageable units. Both benefits are also the main weakness of such a model, as a consequence the feedback loop is stretched to a dangerous length. At the same time the time from feature/technology to market increases, see figure 11.7.

11.5.4 Alternative Re-use Scenario's

Table 11.4 show a number alternative re-use strategies, which have been applied successfully.

11.6 Common Pitfalls

The list of pitfalls in 11.5 has been compiled on the basis of many disastrous or halfway successfull efforts of generic development.

- Spin-out as an independent company (especially for key and base technology)
- Reuse after use (works for good clean designs)
- Opportunistic copy
- Open source
- Inner-source (stimulated open source approach within company scope)
- Evolutionary refactoring (extreme programming)

Table 11.4: *Alternative Re-use Scenario's*

11.7 Acknowledgements

During the first CTT course system architecture, from november 22 until november 26 1999, a lively discussion about generic developments took place, which created a lot of input for this article. I am grateful to the following people, who attended this course: Dieter Hammer, Wil Hoogenstraaten, Juergen Mueller, Hans Gieles, Huib Eggenhuisen, Maurice Penners, Pierre America, Peter Jaspers, Joost Versteijlen, Peter Beelen, Jarl Blijd, Marcel Dijkema, Werner Roelandt, Paul Janson, Ashish Parasrampuria, Mahesh Bandakka, Jodie Ledeboer

I thank Pierre America for working on consistency in spelling and the use of capitols.

- Too generic
- Forced cooperation
- Time platform feature to market
- Unrealistic expectations
- Distance platform developer to customer
- No marketing ownership
- Bureaucratic process (no flexibility)
- New employees, knowledge dilution
- Innovation stops (stable interfaces)
- Underestimation of platform support
- Vulnerability
- Overstretching of product scope
- Non-management, organizational scope increase
- Underestimation of integration
- Component/platform determines business policy
- Subcritical investment

Table 11.5: *Sources of failure in generic developments*

Chapter 12

Product Families Business Analysis and Definition

12.1 Introduction

The creation and evolution of a product family is based on a business analysis. Such a business analyis is used for the definition of the family: which products are member of the family, what distribution of features, which performance range.

Several methods can be used to make the step from business analysis to product family definition, see for instance table 12.1.

- Roadmapping
- Reference Architecture
- Requirements Capturing
- Feature Space Exploration
- Value Engineering
- Scope Determination

Table 12.1: *Methods for Family Analysis*

This article is to be used in the "Family Engineering Handbook", a collective effort of Philips Research employees to consolidate family engineering based experiences.

12.2 Roadmapping

About once per year it is recommended to work for a number of weeks on roadmaps. These roadmaps serve as a shared vision of the next 5 years, see [16]. Roadmapping is done at the level of a product portfolio or product family. The value of roadmapping is that it brings understanding over 5 views: market, product, technology, process and people. This understanding has the time perspective as the main dimension.

The roadmaps provide a time and product portfolio context for the definition of a product family. A number of the activities in roadmapping and product family definition are quite similar; both require an market analysis, a good understanding of commercial opportunities and insight in the technology.

Roadmapping is focused on insight, understanding and shared vision, without any commitment. The definition of a product family results in a more specific detailed output, which is at least partially commital. In other words. Roadmapping is transforming a strategy into tactics, while Product Family definition transforms the tactics into operational activities.

12.3 Reference Architecture

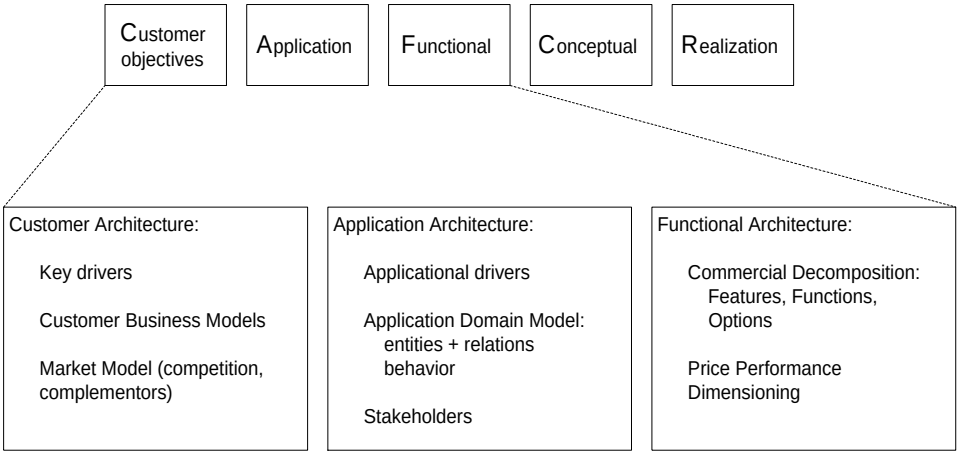


Figure 12.1: Product Family Reference Architecture, zooming in on the views determined by the business analysis and family definition process

A reference architecture covers 5 viewpoints on a product family, see figure 12.1. The business, application and functional architectures are the main subjects of interest during the business analysis and family definition process.

12.3.1 Business Architecture

The business architecture models the world of the customer. Again a number of complementary views are required.

The key-drivers of the customer are identified, see [15]. A limited, but specific set of key-drivers is a powerful guide in the entire creation process.

The business model of the customer is determined, see typical questions addressed by a business model in table 12.2

- Who appreciates what?
- Who pays when for what?
- Who takes decisions?

Table 12.2: *Questions addresses in the business model*

The business of the customer is served by many different suppliers. Some of these suppliers are competing with your own business, while others are complementary. This information is compiled into a market model.

Example Set top boxes are supplied to different kinds of customers, varying from consumers to content providers. In case of the content providers different business models are practiced, ranging from pay-per-view to entirely paid by the advertisers.

The set top box is only a small part of the value chain. Many complementers are active in this entire chain, which starts at the content generation and ends at the television screen of the consumer. Philips is quite active in all complementing products at the consumer side, such as television and video storage, while it is active in parts of the value chain proceeding the set top box.

The competition exists from comparable set top box manufacturers, but also new devices such as game computers (Playstation 2) enter this market.

12.3.2 Application Architecture

The application architecture models the way the user works or enjoys your products in a broader context.

The key-drivers of the business architecture are transformed into application drivers, which describe what the user needs to fulfill the key-drivers of the business. These application drivers provide insight. The direct relation with the key-drivers and the functional requirements provide traceability and a means to focus the requirement process.

Application domain models support the other processes by providing a shared reference. A model describing the entities and their relations ”sets the stage”;

it defines the relevant entities such as persons, tools, deliverables, consumables etcetera. A dynamic view on the application is given in the behavior model. Both models at this level should focus on the main issues, detailed definitions endanger the overview and understanding.

Figure 12.1 explicitly mentions stakeholders as part of the application architecture. Of course stakeholders will show up as an entity. The (human) stakeholders play such an dominant role in the application that it is useful to make a separate overview of the stakeholders and their roles.

Example: An X-ray diagnostic system requires predefined diagnostic procedures to be used easily. These procedures are based on rather specific domain knowledge, such as demographic data, pathology and anatomical data. The essentials of the way of working should be described in the application drivers.

The application model would describe all relevant entities, such as patient, patient table, monitors, UI devices, tube, detector, ECG monitor, film, examination room, technician, nurse, patient etcetera including their relationship.

Note that understanding is the aim of this exercise, not completeness. Those entities and relations should be shown which are relevant for the shared (by commercial and technical people) understanding of the application.

The behavior model would describe the dynamics of the application. It could for instance describe the patient flow, and the information flow.

The application stakeholder view focuses on the human players, which are in this case: referring physician, receptionist, radiologist, patient, technician, nurse, technical support staff of the hospital, etcetera.

12.3.3 Functional Architecture

The functional architecture is the commercial view on the system, describing the commercial flexibility of the products. The functional architecture is the basis of the sales catalog.

The commercial decomposition defines in terms of functions, features and options the capabilities of the products and their structure from commercial point of view. The product manager decides which items to package in sellable products.

The price performance ranges are also defined in the functional architecture.

12.4 ”YoYo-View” over time

To define a product family technical, business and application know how are a prerequisite. Figure 12.2 shows that this know how is often the result of previous experience with single products [1]. The diagram simplifies this learning curve to a

[1] recruitment of experienced people is also an effective way to obtain the know how. In fact the same learning curve is followed, but external.

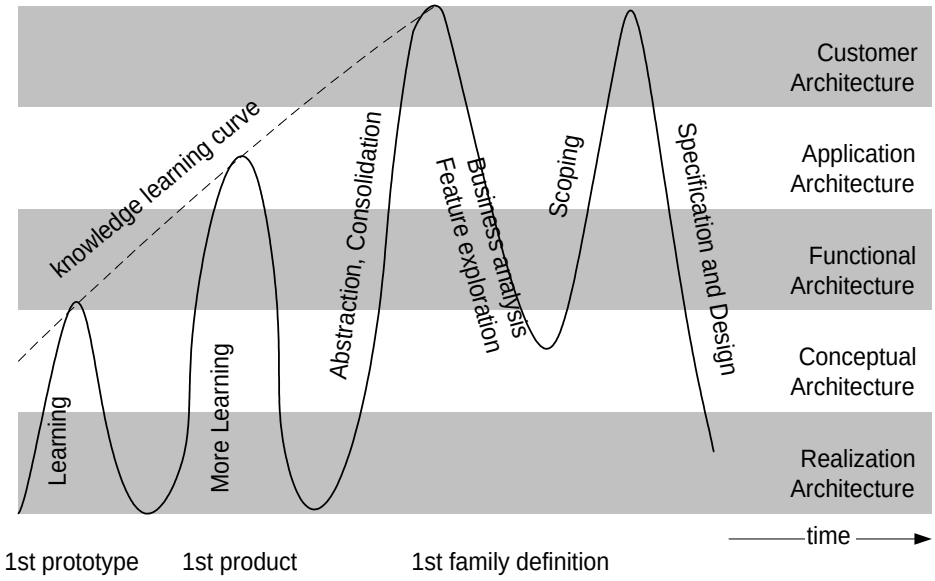


Figure 12.2: The analysis and definition of a family requires a number of iterations over the views in the reference architecture

single prototype and product, in reality more generations are required for the build up of the know how.

When enough knowhow is present in the group of people, this know how is made explicit in the form of a reference architecture. The ”problem” is now analyzed by making a business analysis, feature space exploration and valuation of the features. As shown in figure 12.2 this activity ranges over the business, application and functional architectures.

The next step is to go back to a more fundamental question:

12.5 Relation with the Technical Architecture

The family definition will have to iterate with the technical and implementation architecture. Figure 12.3 shows an example of the contents of a technical architecture in case of a Component Based Product Family.

Rather fundamental decisions which have to be taken for the technical architecture is where to address the requirements, in:

- Product Specific Components,
- Generic Components, or in
- Architecture Guidelines.

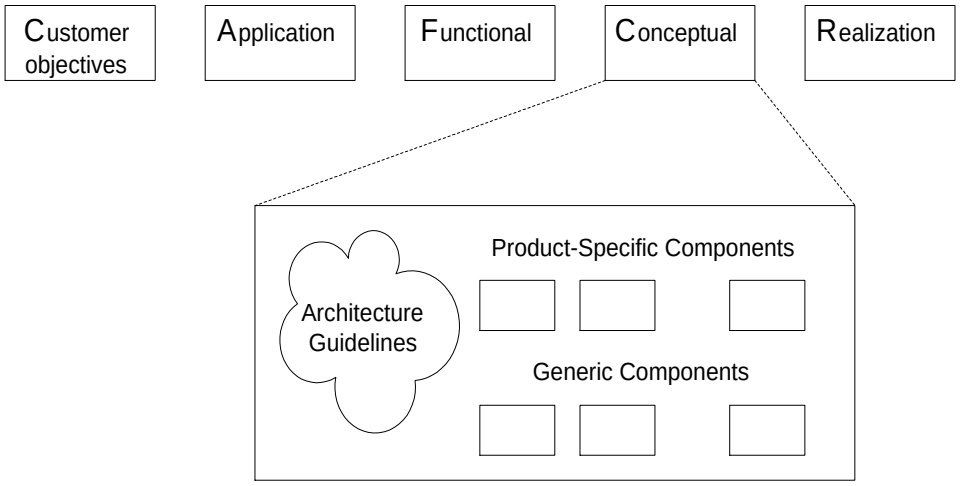


Figure 12.3: Technical Architecture for a Product Family

Ideally the technical structure closely resembles the functional structure, by a natural mapping of functions and features on components.

12.6 Requirements Capturing

Collection and analysis of requirements is indispensable. Many methods exists to do this. In [15] the requirements capturing is described for products. However the methods described in this article are also applicable for Product Families.

Product Family Definition requires special attention for commonality and variation analysis and for product positioning. In section 12.7 some more detailed method is described to address these issues.

- Installation
- Configuration
- Customization
- Lifecycle management (amongst others upgrading)
- Configuration Management
- Licensing strategy

Table 12.3: *Subjects requiring special attention for Product Families*

Also special attention should be paid to the ”introvert” requirements side: the

requirements of the internal stakeholders, such as sales, manufacturing, service etcetera. Table 12.3 shows a list of subjects which require special attention in case of product families.

12.7 Feature Space Exploration and Value Engineering

1. Make an inventory of features
2. Map features on market segments
3. Determine products
4. Map features on products
5. Determine valuation criteria
6. Valuate features per product

Table 12.4: *From Feature Exploration to Valuation per Product*

Analysis of commonality and variation of features over products helps to define the product family in first instance and to make a family design in second instance. This analysis starts with an exploration of the feature space, and results in a valued set of features per product. Table 12.4 shows which steps are taken in this process. See also [15] which describes how to obtain requirements.

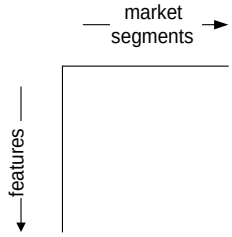


Figure 12.4: Market Feature Map

The features can be mapped on market segments, resulting in a matrix, see figure 12.4. The feature axis can be ordered, for instance by following the key-driver, application driver derivation.

Again iteration is the magic word. Iterate a few times from Market segment to Features and vice versa. If key-drivers are used as structure for the feature axis, then these key-drivers should be included in the iteration. Market segments can have different key-drivers!

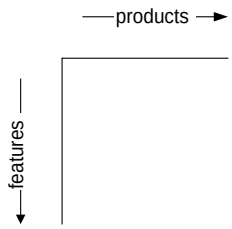


Figure 12.5: Product Feature Map

The Market segmentation can be transformed in products, once sufficient insight is obtained in the market segments and the features involved. This results in a Product Feature Map, see figure 12.5.

- Value 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

Table 12.5: *Example of Valuation Criteria*

Valuation criteria are needed to determine the value of features. Table 12.5 shows an example of Valuation Criteria.

Figure 12.6 shows the result of the entire process. Here all the features have been valuated, the corresponding values are substituted in the matrix.

This matrix is the starting point for the selection process, see section 12.4, which finally have to answer:
Which Feature will be realized When for Which product?

A much more elaborated method for feature space exploration, valuation and scoping can be found in [4].

12.8 Scope Determination

A fundamental question in Product Family approach is the scope of the family
Which part of the Market do we want to serve?

A clear shared answer on this question is the key to an efficient continuation of the Family Creation Process. Some more nuance can be added tot the question by including the time dimension (*When?*).

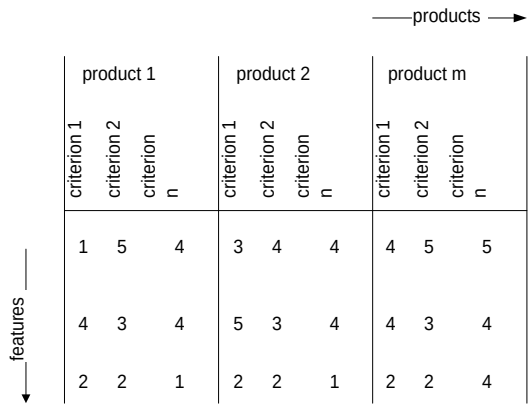


Figure 12.6: Product Feature Map with substituted Numbers

Note that figure 12.2 also simplifies the scoping to a single iteration. In reality some iteration with the technical and implementation architecture takes place.

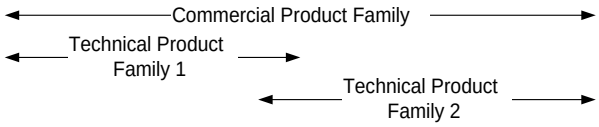


Figure 12.7: Commercial and Technical Viewpoint on Product Families

The scope determination is primarily a commercial scoping. Later in the process, as part of the Family Design, also technical scope determination is needed. Figure 12.7 shows that a commercial Product Family can be realized by two technical product Families.

Example High end products (”Upmarket Televisions”) will emphasize a richness of features, irrespective of for instance memory and processor constraints, while the mid range products (”Mainstream Televisions”) have a severe cost constraint, which translates in memory and processor constraints. From commercial point of view it should appear as one continuous family. From technical point of view the requirements could be conflicting too much, while two technical families with a different optimization focus, match perfectly with the commercial requirements.

12.9 Acknowledgements

Frank van der Linden wrote a position paper on this subject to trigger the discussion for the ”Family Engineering Handbook”. After Frank left Philips Research to join Philips Medical Systems I inherited the job to write this section of the handbook. I thank Frank for writing the original poistion paper which served as a starting point

of this article.

Discussions with Jürgen Müller helped to sharpen the contents of this article. Discussions in the composable architecture meeting, attended by Pierre Ameria, William van der Sterren, Jan Gerben Wijnstra and Jürgen Müller helped to make the article more complete and consistent.

Chapter 13

Role Of Software in Complex Systems

13.1 Why is Software a Bottleneck in Product Development?

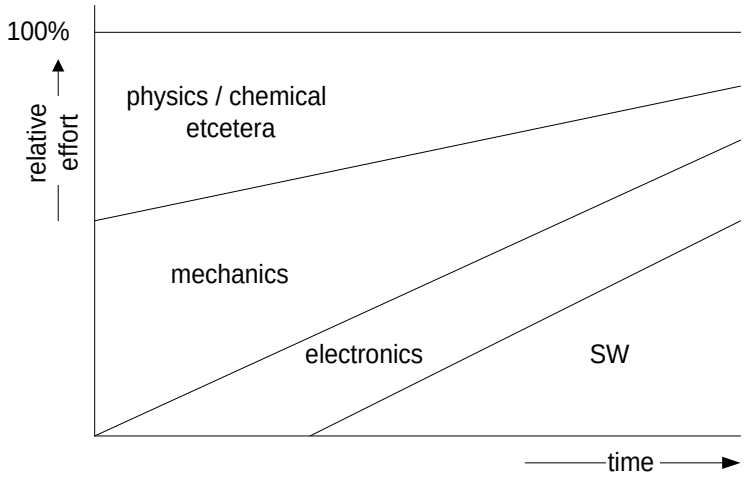


Figure 13.1: The relative contribution of software effort as function of time

The effort of developing software for new products is increasing. Figure 13.1 shows the growth of the SW effort in relation to other technologies.

13.2 System or Software Issues?

Non functional requirements of a system are strongly influenced or even determined by the software architecture. This is also described in [3], which gives one

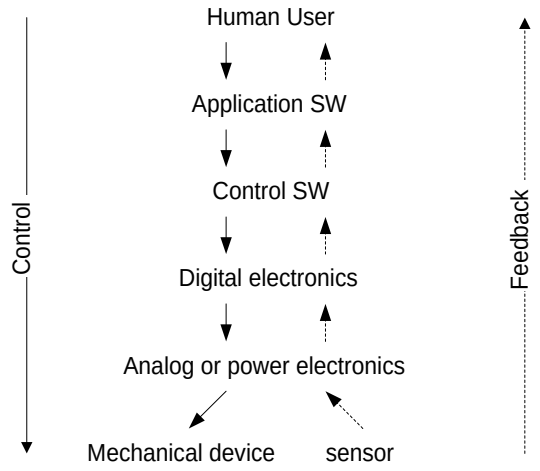


Figure 13.2: The Control Hierarchy of a system along the Technology dimension

particular architecting method to focus on non functional requirements during the software architecture process.

Table 13.1 shows the quality attributes of a system. Table 13.2 shows the performance attributes, while table 13.3 shows the remaining non functional requirements.

All non functional requirements derived from the attributes in table 13.1, 13.2 and 13.3 are system level requirements defining the **what** of the system. In other words the combination of subsystems and technologies together fulfil the non functional requirements.

The System Design decomposes the system in subsystems and implementation technologies. During this step the contribution or the role of a subsystem and implementing technology is determined, for instance by means of aspect design. In [26] and [22] examples are given in the medical and telecommunication domain of quality attributes and aspect design.

Table 13.4 shows the System Level Design Aspects.

Both Quality Attributes and Design Aspects are System Level Issues, however most of these issues are predominantly influenced by the software. The System Architect should: define the system level **what**, co-design the system level **how** and be involved with the single technology or subsystem **how**.

Due to the strong Software impact the software architect should: understand/review the system level **what**, co-design the system level **how** and design the software **how**.

This requires significant domain know-how of the Software Architect, see [9].

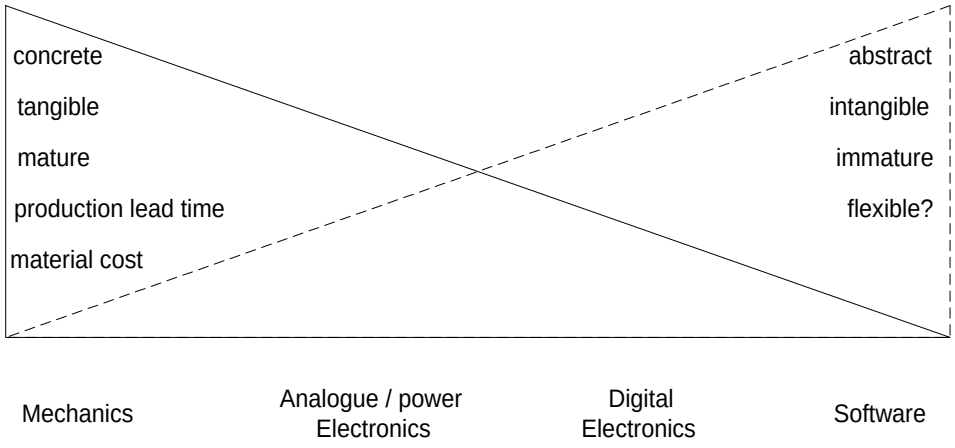


Figure 13.3: Characterization of disciplines, ordered along the level of abstraction

13.2.1 Central versus Local

The design and implementation of system level aspects can be strongly centralized or a large freedom can be left for the more local developers. A well known heuristic is that:

All centralized issues become a bottleneck.

A total anarchy of implementations at the other hand is killing also, especially in the integration phase when conflicting approaches cause havoc.

Software suffers more from conflicting choices, probably due to the lower maturity. Less standards and patterns are obvious, while the technology is still changing fast.

The successfull Software Architect defines the minimum set of design and implementation guidelines,

enabling an integrated system fulfilling the entire set of non functional requirements.

13.3 Acknowledgements

Jürgen Müller helped to sorten out the attributes, aspects, mechanisms etcetera, whcih helps to position the Software Discipline in the System Development.

- Safety
- Security
- Reliability
- Robustness
- Manufacturability
- Testability
- Serviceability
- Configurability
- Installability
- Evolvability
- Portability
- Upgradeability
- Extendability
- Maintainability
- Disposability

Table 13.1: *Quality Attributes for systems*

- Useability
- Appeal, Appearance
- Throughput or Productivity
- Response Time
- Image Quality
- Reproduceability
- Predicatability
- Accuracy

Table 13.2: *Performance Attributes for systems*

- Cost price
- Cost of operation
- Interaction with environment
- Power consumption
- Consumption rate (water, air, chemicals, etcetera)
- Size, weight
- Resource utilization
- Logistics flexibility
- Lead time
- Standards Compliance

Table 13.3: *Non Functional Attributes, not yet mentioned in Quality and Performance Attributes*

- Philosophy per performance, quality or other attribute
- Granularity, Scoping, Containment, Cohesion, Coupling
- Interfaces, Allocation, Budgets
- Information Model (Entities, Relations, Operations)
- Identification, Naming
- Static Characteristics, Dynamic Behavior
- System Level Infrastructure
- Software Development Process, Environment, Repository and Tools
- Feedback tools, for instance Monitoring, Statistics and Analysis
- Persistence
- Licensing, SW-keys
- Initialization, Start-up, Shutdown
- Set up sequence
- Technology Choices
- Make, Outsource, Buy or Interoperate decisions

Table 13.4: *System Design Aspects*

- Error Handling, Exception Handling, Logging
- Process, Tasks, Threads
- Configuration Management; Packages, Components, Files, Objects, Modules, Interfaces
- Automated Testing: special Methods, Harness, Suites
- Signalling, Messaging, Call Back scheduling, Notification, Active Data, Watchdogs, Time-outs
- Locking, Semaphores, Transactions, Checkpoints, Deadlock Detection, Role Back
- Identification, Naming, Data Model, Registry, Configuration Database, Inheritance, Scoping
- Resource Management, Allocation, Fragmentation Prevention, Garbage Collection
- Persistence, Caching, Versioning, Prefetching, Lazy Evaluation
- Licensing, SW-keys
- Bootstrap, Discovery, Negotation
- Call graphs, message tracing, object tracing, etcetera
- Distribution, Allocation, Transparancy; Component, Client / Server, Multi tier model

Table 13.5: *List of Software Mechanisms, which are frequently applied to solve the system level design aspects*

Chapter 14

Intermezzo: The Tense Relation between Architect and Manager

14.1 Introduction

This intermezzo is generalizing the architect and the manager. No architect nor manager will exactly look like the generalization in this intermezzo. The generalization should help to understand the tension in the relationship between both groups.

14.2 What is a manager, which managers are addressed here?

A manager is someone who manages everything needed to get a task executed. The manager has the responsibility for the task and gets the required authority to do it. Every Process in the simplified business decomposition in [12] generally has a manager associated with it who is responsible for the execution of that process. Often these tasks are further decomposed and there are managers associated with the sub-processes as well.

Examples of managers which the system architect frequently encounters are shown in figure 14.1.

14.3 Comparison of Architect and Manager

Managers have a well defined responsibility, related to their function. In most organizations managers also are empowered accordingly. The scope of responsibility is limited, the total responsibility is divided over many managers.

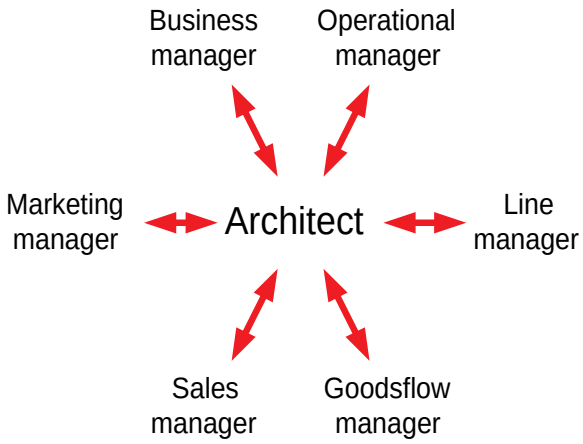


Figure 14.1: Managers which frequently interact with architects

The responsibility of the architect is much more fuzzy. For every aspect he is working on there is some manager which has the formal responsibility for that specific subject. The architect has a limited formal power. At the other hand his informal influence is huge.

	Architect	Manager
scope	wide	limited
formal weight	low	high

Table 14.1: *Comparison of Responsibilities*

Table 14.1 summarizes the comparison of responsibilities between architect and manager.

The view on solutions, summarized in table 14.2 is quite different. The architect partially trusts his intuition and has the notion of an elegant solution. The word elegant can cover many aspects, such as: balanced, simple, beautiful. As representative of the stakeholders he will guard the fitness for use, is it the "right" solution? At the same time the architect will place the solution in a time perspective, is the solution "futureproof"?

Most managers stay close to their task and responsibility. A solution which matches the specification is per definition good. If there are no complaints, there is no problem.

The manager is control minded, he likes to be in control of the task he is performing, which is exactly his task. Often this being control minded is projected on his environment, which results in being conformance demanding. The people in his environment should conform to the way the manager wants to perform his

	Architect	Manager
design	elegant	if it works it is OK
application	perfect fit	no complaints
futureproof	important	task dependent

Table 14.2: *View on Solution*

task.

The architect has an entirely different personality. To be able to act as representative of the stakeholders he needs independence and curiosity. At the same time he needs to be critical, is this the best way to do it?

Table 14.3 summarizes these personal characteristics.

Architect	Manager
Independent	Conformance Demanding
Critical	Control minded
Curious	

Table 14.3: *Personal Characteristics*

The architect is a significant part of his time involved in the turbulent outside world, inhabited with demanding customers in changing markets with aggressive inventive competitors, and innovative suppliers. At the same time he is active in the company across many internal boundaries, which enables him to detect, analyze and to help solving many internal problems. This dynamic overview of outside and internal world all the time requires changes. An architect sees changes as a fact of life.

The limited scope and the heavy weight of the responsibility of their task results in an opposite viewpoint in the case of managers. Their experience is that changes always introduce problems, involve uncertainties, and trigger more changes. The resulting behavior is to avoid changes. See table 14.4.

	Architect	Manager
viewpoint	changes needed due to: • stakeholders • time • problem analysis	changes introduce: • problems • uncertainties • new changes
attitude	fact of life	avoid changes

Table 14.4: *Changes, viewpoint and attitude*

The conventional hierarchical view on organizations create the expectation that the manager determines the direction. In reality most managers equate their task setting to the direction, which means that the direction has a rather limited horizon.

The architect has a broad perspective and knowhow, while (good) architects also have vision. This is a natural combination to provide true leadership.

Some architects are handicapped by an introvert personality which makes it difficult to "sell" the vision and to take the leaders position. It will be clear that teamwork of manager and architect will work wonders in such a case.

Architect	Manager
provides direction based on know how and vision	direction based on task setting
	"title" creates expectation of direction setting

Table 14.5: *Leadership aspects*

The personal ambition of managers and architects are opposite as well, see table 14.6. Many managers are driven by normal career incentives: higher position, power, status and more money. Architects seem to be driven by the case at hand, they want to achieve the "best" solution.

Architect	Manager
best solutions	highest hierarchical level

Table 14.6: *Ambition*

This difference in ambition makes the architect difficult to control, because he is rather insensitive for the normal means of control, such as promotions and salary raises.

14.4 How to improve the relationship

The starting point for any solution is the recognition of the problem. This intermezzo is primarily written to create awareness of the problem, no golden bullet solution will be given here.

The direction which is quite promising to address this problem is modern management techniques:

- Empowerment
- Delegation
- Leadership instead of task driven management

- Process orientation instead of hierarchical organizations
- Teamwork
- Mutual Respect
- Recognition of diversity and nonconformity
- Reverse Appraisal

The architect plays a vital role in bootstrapping these management techniques. In many techniques the architect plays the role of catalyst due to the combination of personal characteristics such as independence and know how. If the architect hides in technological solutions, the required change is blocked.

14.5 Acknowledgements

Jürgen Müller attended me on the fact that telling only the negative (The relationship is tense) is not good enough. An architect should always look for the constructive way out. I therefor added section 14.4. Wil Hoogenstraaten also commented that the described relationship is well recognizable, but how to escape from this situation?

Chapter 15

Case Study: Medical Imaging; From Toolbox to Product to Platform

15.1 Introduction

The Medical Imaging workstation was an early large scale Object Oriented product. Originally intended to become a re-useable set of toolboxes, it evolved in a family of medical workstations and servers.

This article describes the evolution from different viewpoints, to serve as background material for a number of case studies of the Gaudí project.

15.2 Product Context

15.2.1 Philips Medical Systems

Philips Medical Systems is a major player in the medical imaging market. The main competitors are GE and Siemens. The Product Creation focus of Philips Medical Systems is modality oriented, as shown in figure 15.1.

The common technology in conventional X-ray systems is developed by component oriented business groups, which make generators, tubes, camera's, detectors, etcetera. The so-called "System-groups" have a more clinical focus, they create the clinical oriented systems on the basis of the common available components.

The non X-ray groups[1] mainly build large complex general purpose imaging equipment. The imaging principles in CT and MR are less direct, which means

[1] A poor name for this collection; The main difference is in the maturity of the modality, where this group exists from relative "young" modalities, 20 a 30 years old.

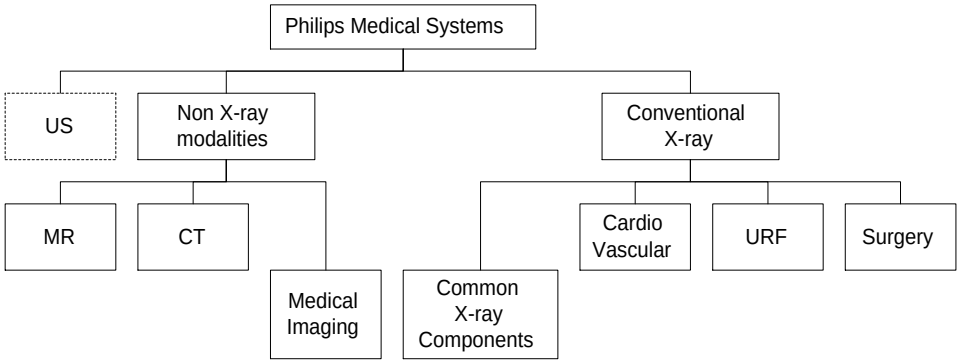


Figure 15.1: Philips Medical Systems, schematic organization overview.

that an image reconstruction step is required after acquisition to form the viewable images.

Ultra Sound (ATL) is acquired by Philips Medical Systems recently. It is not fully integrated in the organization.

The main markets of Philips Medical Systems are radiology and cardiology, with a spin off to the surgery market.

15.2.2 Radiology

Traditionally the radiologist makes and interprets images from the human body. A referring physician requests an examination, the radiologist responds with a report with his findings. Figure 15.2 shows a generic set of Radiology drivers.

Philips Medical Systems core is the imaging equipment in the examination rooms of the radiology department[2]. The key to useful products is the combined knowledge of application (**what**) and technology (**how**).

15.3 Historic Phases

The development model of Medical Imaging has changed several times. Roughly the phases in table 15.1 can be observed. The first phase can best be characterized as technology development, with poor Market and Application feedback. The next phase overcompensates this poor feedback by focusing entirely on a product.

Philips Medical Systems has been striving for re-useable viewing components at least from the late seventies. This quest is based on the *assumption* that the viewing of all Medical Imaging Products is so similar, that *cost reduction* should

[2] equally important core for Philips Medical Systems is the cardio imaging equipment in the catherization rooms of the cardiology department, which is out of the Medical Imaging Workstation scope.

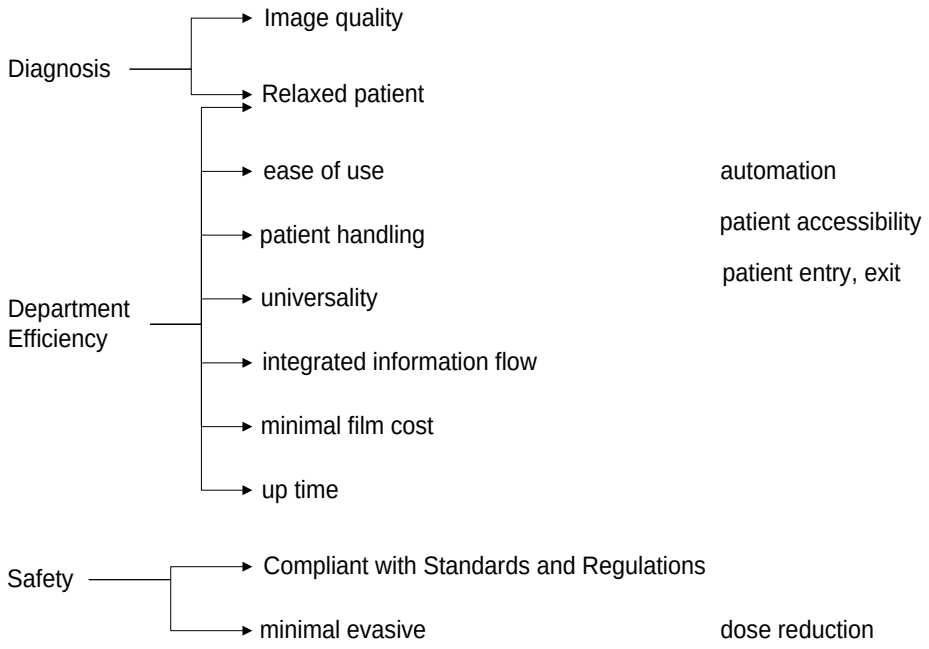


Figure 15.2: Generic drivers of Radiology Departments

be possible when a common implementation is used. The lessons learned during this long struggle have been partially consolidated in [14].

The group of people, which started the Common Viewing development, applied a masive amount of technology innovations, see table 15.2.

15.3.1 Basic Application and Toolboxes

The goal of the common viewing development was to create an extensive set of toolboxes, to be used for viewing in all imaging products. The developers of the final products had fine-grain access to all toolboxes. This approach is very flexible and powerful, however the penalty of this flexibility is that the integration is entirely the burden of the product developer.

The power of the toolboxes was demonstrated in a **Basic Application**. This basic application was a superset of all available features and functions. From clinical point of view a senseless product, however a good vehicle to integrate and to demonstrate.

Figure 15.3 shows the idealized layering of the toolboxes and the the Basic Application in september 1991. the toolbox layer builds upon the Sun computing platform (Workstation, the Sun version of UNIX SunOS and the Sun windowing environment Sunview). The core of common viewing is the imaging and graphics

- 1987-1991 Advanced Development (”Common Viewing”), result: Basic Application plus toolboxes
- 1991-1992 Development of 1^{st}product: Medical Imaging R/F
- 1992-1994 Parallel Development of 2^{nd} product: Medical Imaging CT/MR
- 1994-1997 Family Development
- 1997-2000 Transformation in re-useable components

Table 15.1: *Phases of Medical Imaging*

- Standard UNIX based workstation
- Full SW implementation, more flexible
- Object Oriented design and implementation (Objective-C)
- Graphical User Interface, with windows, mouse etcetera
- Call back scheduling, fine-grained notification
- Data base engine, fast, reliable and robust
- Extensive set of toolboxes
- Property based configuration
- Multiple coordinate spaces

Table 15.2: *Technology innovations introduced by the initial developers of Common Viewing*

toolbox, and the UI gadgets and style.

15.3.2 Medical Imaging X-Ray

Figure 15.4 shows the X-ray rooms which are involved from the examination until the reading by the radiologist. Around 1990 the X-ray system controls were mostly in the control room, where the operator of the system performed all settings from acquisition setting to printing settings. Some crucial settings can be performed in the room itself, dependent on the application. The hardcopies were produced as literal copies of the screen of the monitor. The printer was positioned at some non-obstrusive place.

The consequence of the literal screen copy was that a lot of redundant infor-

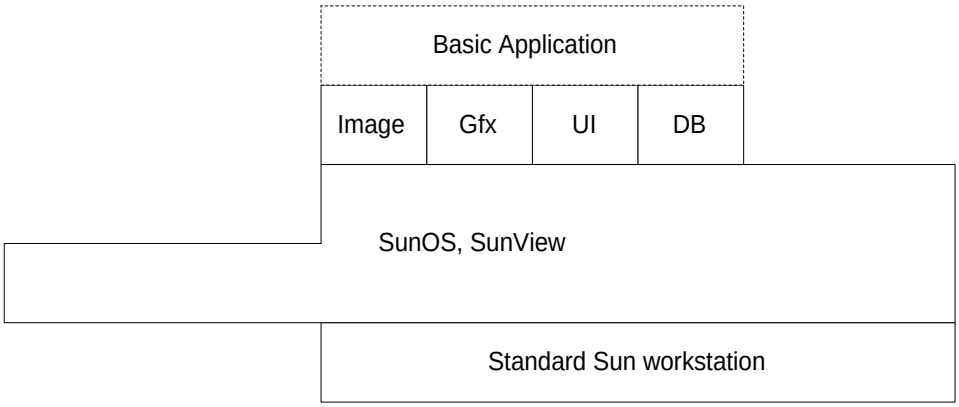


Figure 15.3: Idealized layering of SW toolboxes and Basic Application in september 1991

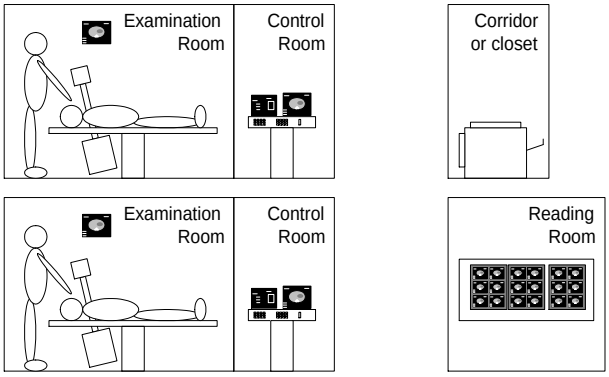


Figure 15.4: X-ray rooms from examination to reading around 1990

mation is present on the film, such as patient name, birth date and acquisition settings. On top of that the field of view was supposed to be square or circular, although the actual field of view is often smaller due to the shutters applied.

The economic existence of Medical Imaging X-ray was based in 1992 on improvements of this printing process. The patient, examination and acquisition information is orderly shown in one viewport, removing all the redundant information near the images itself. A further optimization is applied by a *fit-to-shutter* formatting. These 2 steps together reduce the film use by 20% to 50%.

The user actions needed for the printing are reduced as well, by providing print protocols, which perform the repetitive activities of the printing process. The effectiveness of this automation depends strongly on the application, some applications require quite some fine-tuning of the contrast-brightness, or an essential selection step, which require (human) clinical knowhow.

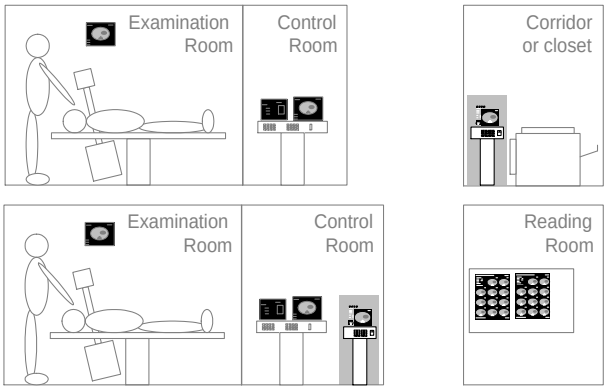


Figure 15.5: X-ray rooms from examination to reading, when Medical Imaging is applied as printserver

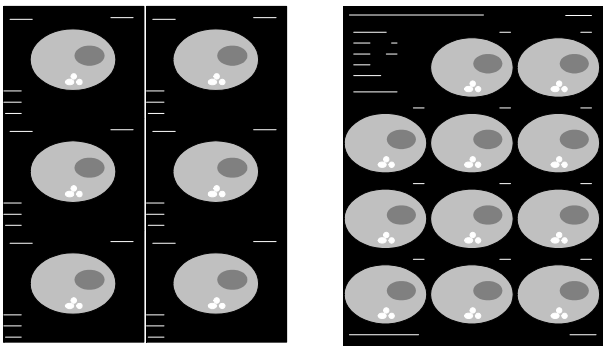


Figure 15.6: Comparison of convential *screen copy* based film and a film produced by Medical Imaging. This case is very favourable for the Medical Imaging approach, typical gain is 20% to 50%.

A prominent sales feature at conferences was the 9-button remote control. The elementary viewing functions, such as patient/examination selection, next/previous image and contrast/brightness. This remote control lowered the threshold for clinical personnel, both radiologist as well as technical, enough to catch their interest: The Medical Imaging was not sold as a disgusting computer or workstations, rather it was positioned as a clinical appliance.

The definition of the Medical Imaging was done by marketing, which described that job as a luxury problem. Normally heavy negotations were required to get features in, while this time most of the time marketing wanted to reduce the (viewing and user interface) feature set, in order to simplify the product.

From software point of view the change from basic application to clinical product was tremendous. The grey areas in figure 15.7 indicate new SW. The

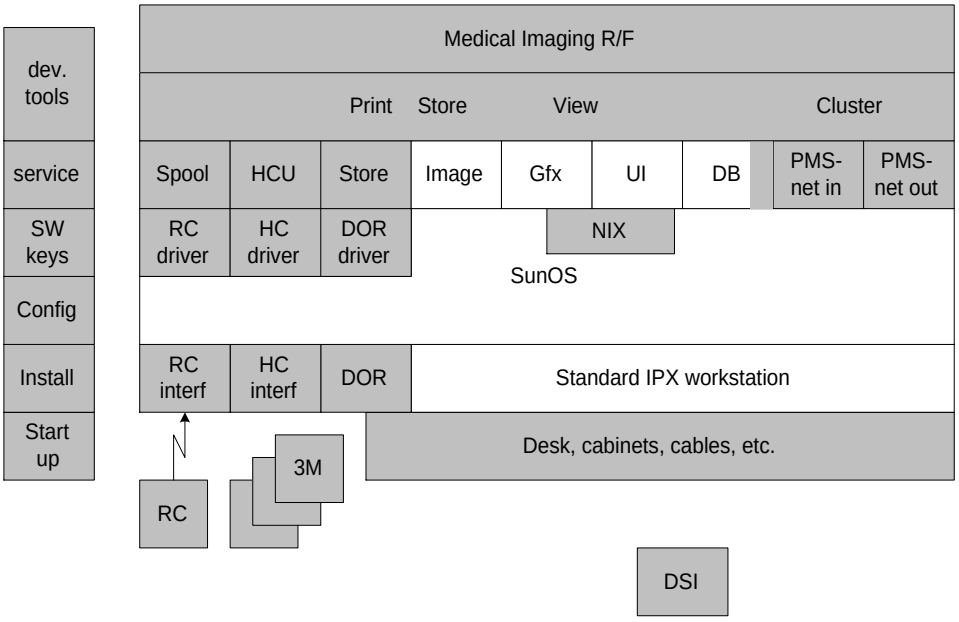


Figure 15.7: Idealized layers of the Medical Imaging R/F software in september 1992

amount of code increased from 100 klines to 350 klines of code.

15.3.3 Second Concurrent Product: Medical Imaging CT/MR

Upto 1992 the Medical Imaging organization had a single focus, first on toolboxes, later on Medical Imaging R/F. In 1993 it was decided to apply the Medical Imaging also on CT and MR.

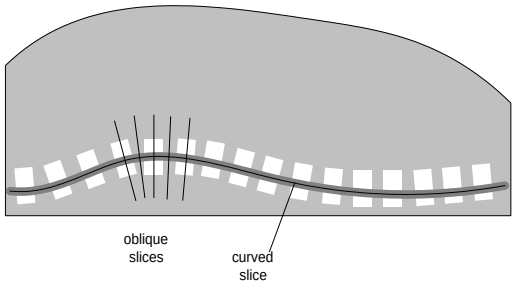


Figure 15.8: Example of Multi Planar Reformatting applied on the spine

The printing functionality of CT and MR scanners improves significantly when Medical Imaging is applied as printserver. However the CT and MR applications

can benefit also from interactive functionality, more than the X-ray applications. An clear example is the Multi Planar Reformatting (MPR) functionality, where arbritary slices are reconstructed from the volume data set.

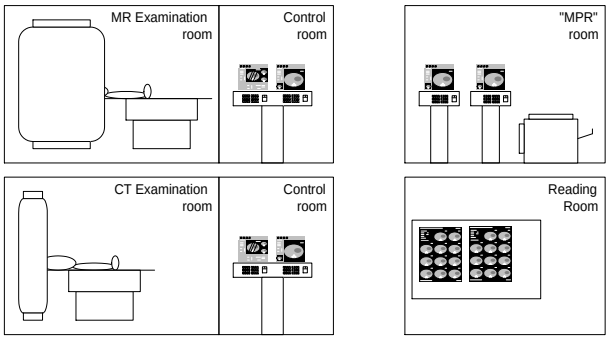


Figure 15.9: Example of CT and MR department, where Medical Imaging is deployed

Superficially X-ray viewing looks the same as CT and MR viewing. However the viewing is different in many subtle ways. A fundamental difference is that X-ray images are *projection* images, while CT and MR images are *slices*, which means that CT and MR images have a 3D ”meaning”, which is missing in X-ray images. The 3D relationship is amongst others used for navigation, a *point-and-click* type of user interface: clicking on a scanogram immediately shows the related slice(s) at that position.

The greylevel mapping for these modalities is performed in technical terms by means of a clipped linear mapping. From implementation point of view the difference in user perception between contrast/brightness for X-ray images (angle and offset of the linear mapping) versus the window width/window level for CT and MR images was totally underestimated.

Table 15.3 shows the differences between the images of these modalities. The combination of different image characteristics and different clinical application propagates into the specification and design. Table 15.4 shows a list of differences in the specification caused by the differences in table 15.3.

The software was significantly extended, the code size increased from 350 klines to 600 klines. Note that this is not only an extension with 250 klines, from the original 350 klines roughly half was modified or removed. In other words a significant amount of refactoring has taken place concurrent with the application extensions. Figure 15.10 shows the (idealized) SW structure at the completion of Medical Imaging CT/MR and the second release of Medical Imaging R/F. Light grey blocks represent new code, dark grey represents major redesigns.

All diagrams 15.3, 15.7 and 15.10 are labelled as *idealized*. This adjective is used because the actual software structure was less *well structured* than presented

	X-ray	CT	MR
image	projection	slice	slice
structure	single image	stack	stack
	or time series	or volume	or more complex
greylevel mapping	contrast	window width	window width
	brightness	window level	window level
resolution	1024^2	512^2	256^2
contrast noise ratio	10 bit	12 bit	8 bit
value		absolute	acquisition dependent

Table 15.3: *Differences between X-ray, CT and MR images*

by these diagrams. Part of the refactoring in the 1992-1994 time frame was a cleanup, to obtain well defined dependencies between the software-"groups". These groups were more fine-grained than the blocks in these diagrams.

15.3.4 Towards Workflow

Medical Imaging R/F and Medical Imaging CT/MR were psotioned as *modality enhancers*. The use of these systems enhances the value of the modality. They are used in the immediate neighborhood of the modality, before the reporting is done. From sales point of view these Medical Imagings are additional options for a modality sales.

The radiology workflow is much more than the acquisition of the images. Digitalization of the helathcare information flow requires products which fit in the broader context of radiology and even the diagnostic workflow. Figure 15.11 shows the competitive positioning of Medical Imaging in 1995, and the positioning of a new class of Medical Imaging products which focus more on workflow added value.

15.4 Process and Organization

15.4.1 Common Viewing

Common Viewing was a self sustained group, reporting to and financed directly by the PMS management. Somehow this group collected creative and rather self-willed individuals, which determined their own course. This is reflected by the technology choices (see table 15.2), but also by the processes and organization.

To a certain degree the culture is similar to Extreme Programming [1], such

- viewing and print preparation
  - navigation support
  - multi-image view
  - greylevel control
- specialized clinical functions
  - vascular and cardio analysis (X-ray)
  - dental (CT)
- print protocols
- information model

Table 15.4: *Specification differences caused by modality differences*

as short iteration cycles and peer programming. If this book had been published ten years earlier it would have been used by this group for sure, which would have helped them amongst others in getting a better application focus and more regression testing.

15.4.2 Medical Imaging R/F

The common viewing department was combined with the "DSI" [3] development team to form the Common Digital Systems (CDS) department. CDS was formally part of the X-ray product group.

This combination eased the development of Medical Imaging R/F, because both sides of the interface were developed within the same organizational entity.

Two entirely different cultures were merged here in one organization. In practice it remained two separate groups under a single management team.

15.5 Acknowledgements

Hans Brouwhuis reviewed the article, providing valuable feedback with respect to the reader viewpoint.

[3] DSI is the image processing chain and user interface of the URF X-ray systems. It is a very focused design, fitting in the right price performance points for the cost sensitive URF market

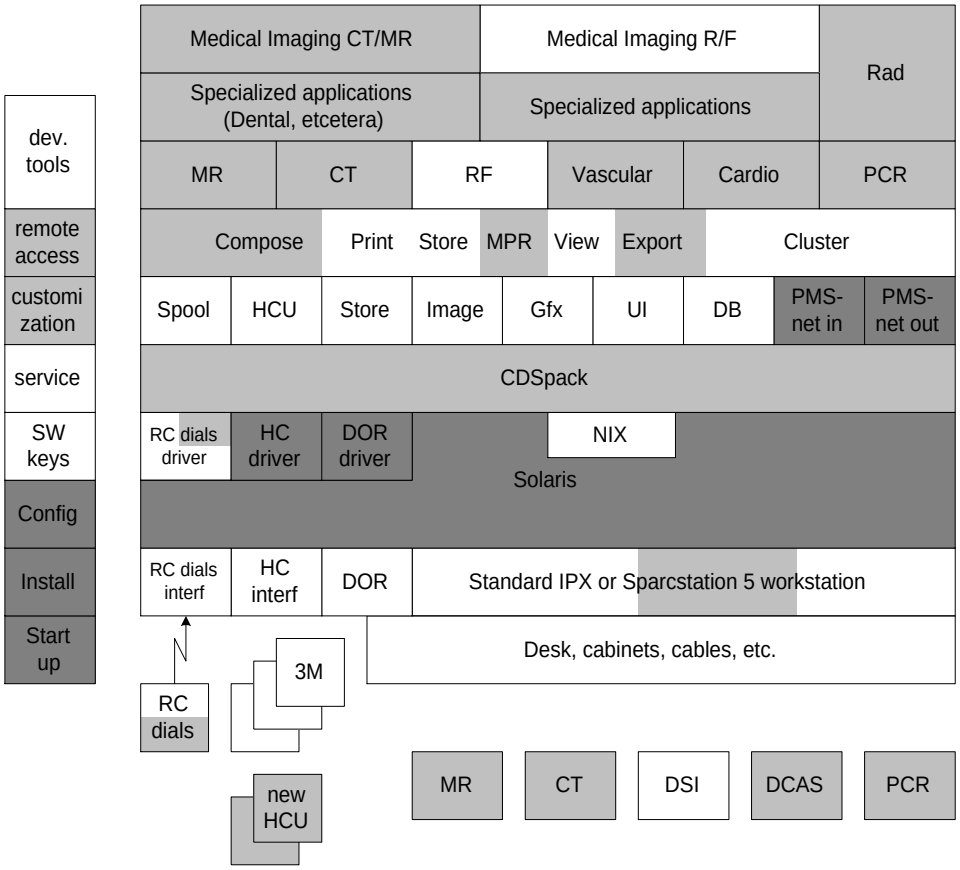


Figure 15.10: Idealized layers of the Medical Imaging software in june 1994

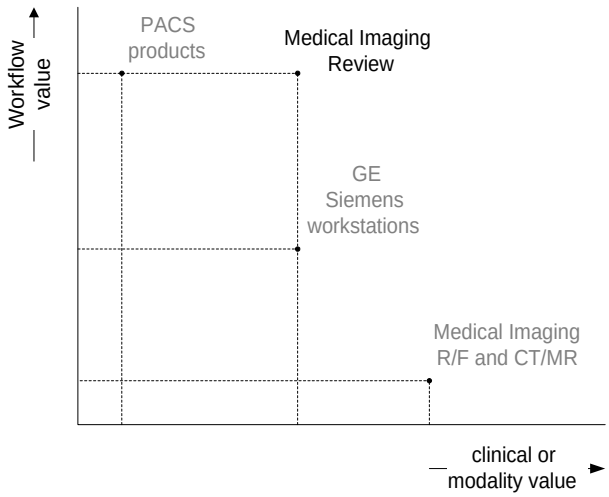


Figure 15.11: Competitive positioning of Medical Imaging, existing products and potential products

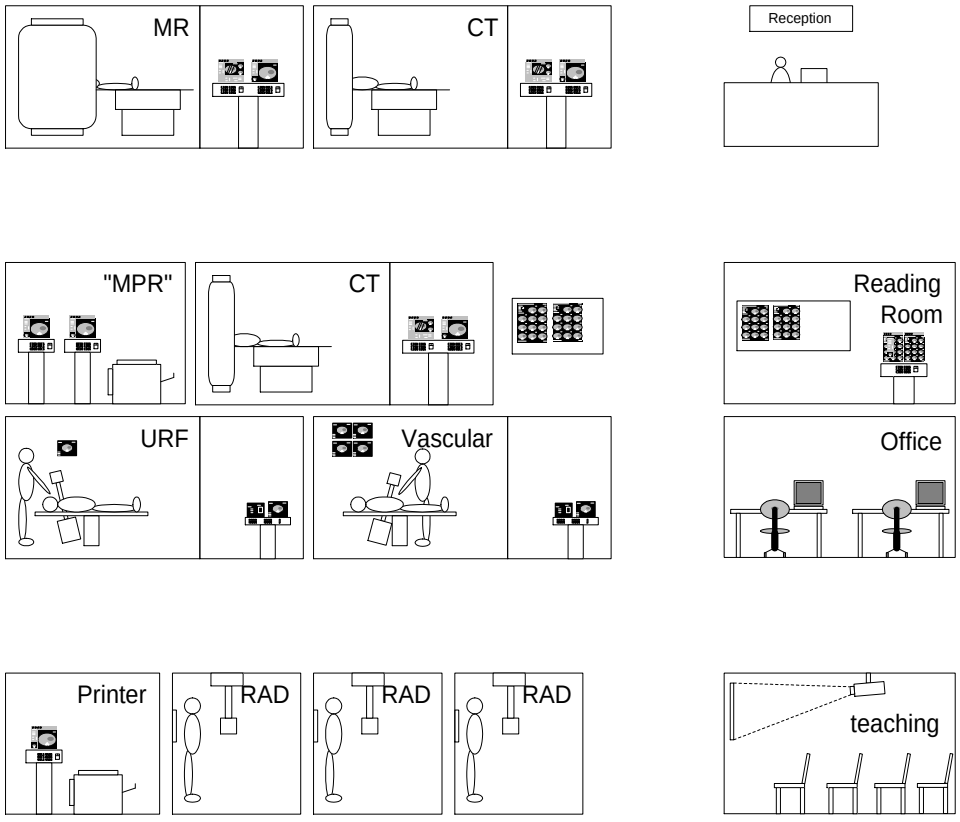


Figure 15.12: Radiology department as envisioned in 1996

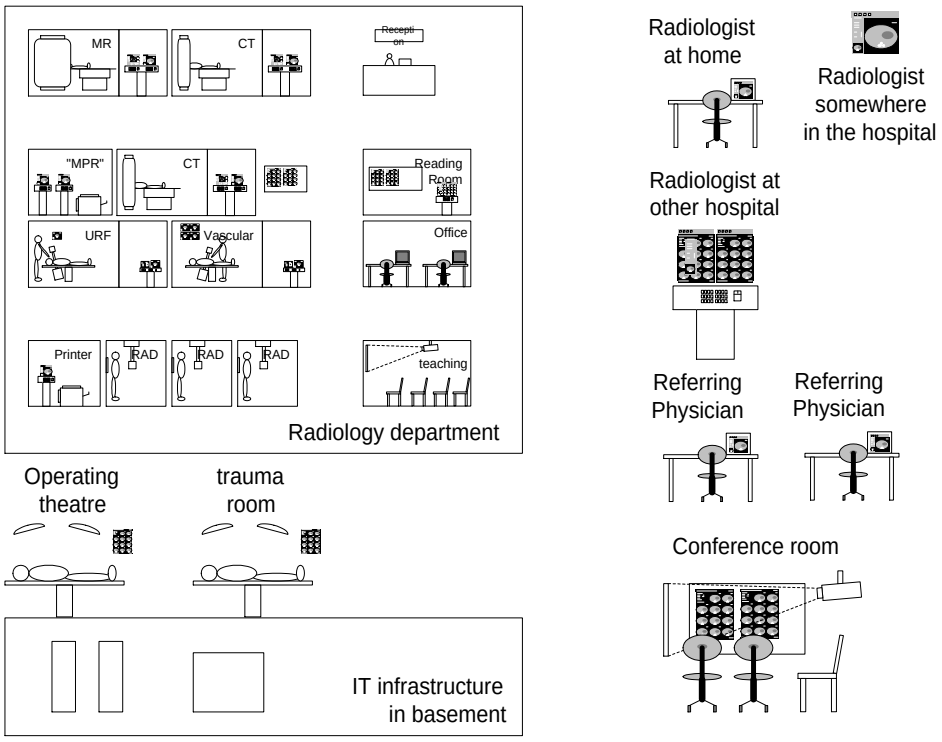


Figure 15.13: Medical Imaging in healthcare workflow perspective, as envisioned in 1996

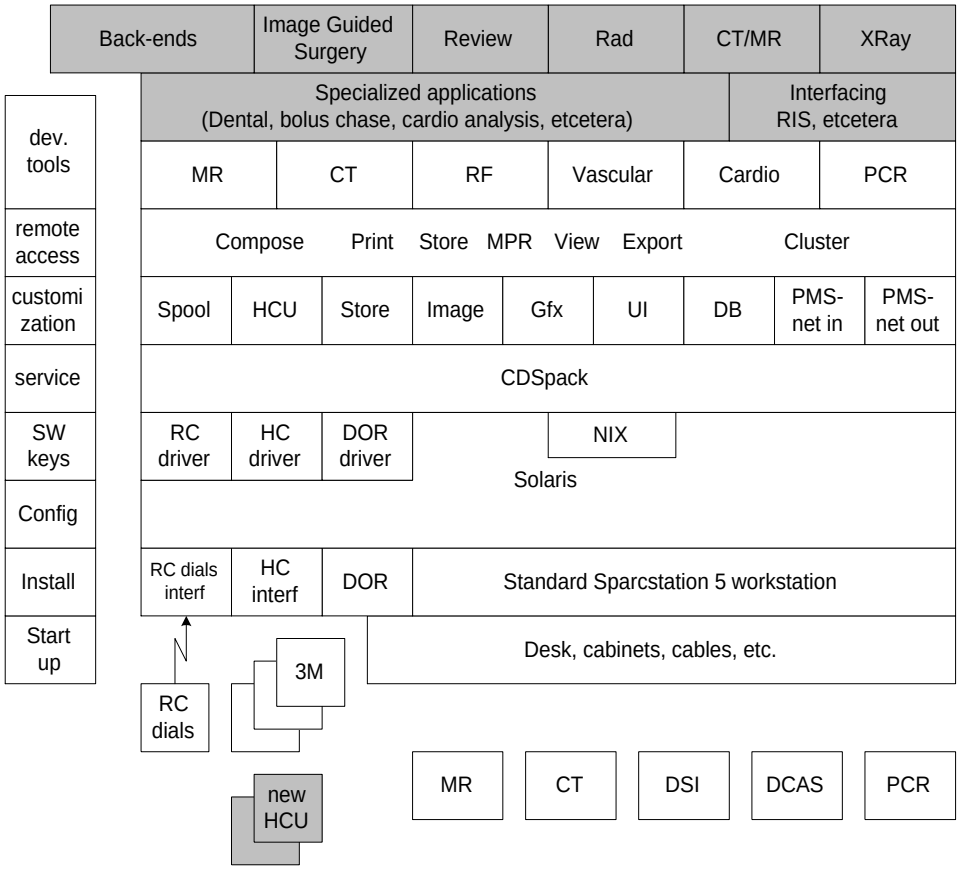


Figure 15.14: Idealized layers of the Medical Imaging software in 1996

Bibliography

[1] Kent Beck. *Extreme Programming Explained; Embrace Change*. Addison-Wesley, Reading, MA, 2000.

[2] B.W. Boehm. A spiral model of software development and enhancement. *IEEE Computer*, May 1988.

[3] Jan Bosch and Peter Molin. Software architecture design: Evaluation and transformation. In *1999 IEEE Engineering of Computer Based Systems Symposium*. IEEE Computer Based Systems, 1999. This article describes an architecting method focused on non functional requirements.

[4] Jean-Marc DeBaud and Klaus Schmid. A systematic approach to derive the scope of software product lines. In *21^{st} inernational Conference on Software Engineering; Preparing for the Software Century*, pages 34–47. ICSE, 1999.

[5] IFIP. *Software Architecture; TC2 First Working IFIP Conference on Software Architecture (WICSA1)*, 1999.

[6] Ivar Jacobson, Grady Booch, and James Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, Reading, MA, 1999.

[7] Ivar Jacobson, Martin Griss, and Patrik Jonsson. *Software Reuse; Architecture, Process and Organization for Business Success*. ACM Press, New York, 1997.

[8] Klaus Kronlöf, editor. *Method Integration; Concepts and Case Studies*. Wiley, Chichester, England, 1993. a useful introduction is given in Chapter 1, The Concept of Method Integration.

[9] Philip Kruchten. The software architect- and the software architecture team. In *Software Architecture; TC2 First Working IFIP Conference on Software Architecture (WICSA1)* [5], pages 565–583. This article describes required skills for architect and architecture team; traps and pitfalls; Personality profile based on Myers-Briggs Type Indicator.

[10] James N. Martin. *Systems Engineering Guidebook*. CRC Press, Boca Raton, Florida, 1996.

[11] Gerrit Muller. The arisal of the system architect. /www.extra.research.philips.com/natlab/sysarch/index.html, 1999.

[12] Gerrit Muller. Positioning the system architecture process. /www.extra.research.philips.com/natlab/sysarch/index.html, 1999.

[13] Gerrit Muller. The product creation process. /www.extra.research.philips.com/natlab/sysarch/index.html, 1999.

[14] Gerrit Muller. Product families and generic aspects. /www.extra.research.philips.com/natlab/sysarch/index.html, 1999.

[15] Gerrit Muller. Requirements capturing by the system architect. /www.extra.research.philips.com/natlab/sysarch/index.html, 1999.

[16] Gerrit Muller. Roadmapping. /www.extra.research.philips.com/natlab/sysarch/index.html, 1999.

[17] Gerrit Muller. The system architecture homepage. /www.extra.research.philips.com/natlab/sysarch/index.html, 1999.

[18] Gerrit Muller. The importance of feedback for architecture. /www.extra.research.philips.com/natlab/sysarch/index.html, 2000.

[19] Gerrit Muller. Process decomposition of a business. /www.extra.research.philips.com/natlab/sysarch/index.html, 2000.

[20] Gerrit Muller. The system architecture process. /www.extra.research.philips.com/natlab/sysarch/index.html, 2000.

[21] Gerrit Muller. What is a process? /www.extra.research.philips.com/natlab/sysarch/index.html, 2000.

[22] Jürgen Müller. Aspect design with the building block method. In *Software Architecture; TC2 First Working IFIP Conference on Software Architecture (WICSA1)* [5], pages 584–601. /nlww.natlab.research.philips.com:8080/research/swa_group/mueller/bb_stuff/aspect_article/aspect.pdf.

[23] Eric S. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly, 1999.

[24] Eberhardt Rechtin and Mark W. Maier. *The Art of Systems Architecting*. CRC Press, Boca Raton, Florida, 1997.

[25] Carnegie Mellon Software Engineering Institute SEI. Software engineering management practices. /www.sei.cmu.edu/managing/managing.html, 2000.

[26] Jan Gerben Wijnstra. Quality attributes and aspects of a medical product family. submitted to the Software Track of the HICSS-34, January 2001 URL: http://nlww.natlab.research.philips.com:8080/research/swa_group/wijnstra/ExternalPublications/hicss34/HICSSPaperJGW.pdf, 2000. This article describes the flow of non functional requirements to (software) implementation.

History

Version: 0.5, date: June 22, 2001 changed by: Gerrit Muller

- corrected part of missing or wrong references and citations

Version: 0.4, date: February 6, 2001 changed by: Gerrit Muller

- Added chapter "Function Profiles; The sheep with 7 legs"

Version: 0.3, date: November 1, 2000 changed by: Gerrit Muller

- Added chapter "Role and Task of the System Architect"
- Added chapter "Role of Software in Complex Systems"

Version: 0.2, date: March 29, 2000 changed by: Gerrit Muller

- Added chapter "Product Families Business Analysis and Family Definition"

Version: 0.1, date: March 24, 2000 changed by: Gerrit Muller

- Added Preface; System Architecture: The Golden Bullet?

Version: 0, date: March 21, 2000 changed by: Gerrit Muller

- Created very preliminary bookstructure from available Gaudí articles and intermezzo's, no changelog yet