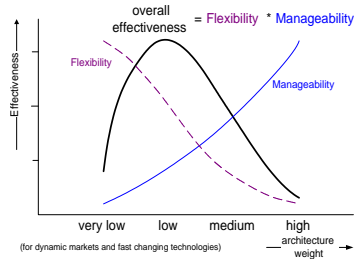


Light Weight Architecture: the way of the future?

-



Gerrit Muller

Buskerud University College

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

`gerrit.muller@embeddedsystems.nl`

Abstract

The question : "What is an architecture" is addressed. Trends in the customer world and in the technology are used to obtain an outline of the product requirements. The customer world itself is a value chain consisting of quite heterogeneous stakeholders.

To satisfy the needs of these customers an integral approach is required. Architectures play a key role in such an integral approach.

Architecture lessons from practice are given to illustrate criteria for a good architecture are discussed. The concept of architecture-weight is introduced.

Distribution

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

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

version: 2.3

status: finished

February 10, 2011

1 Introduction

Architecture is the combination of the know how of the solution (technology) with understanding of the problem (customer/application). The architect must play an independent role in considering all stakeholders interests and searching for an effective solution. The fundamental architecting activities are depicted in figure 1.

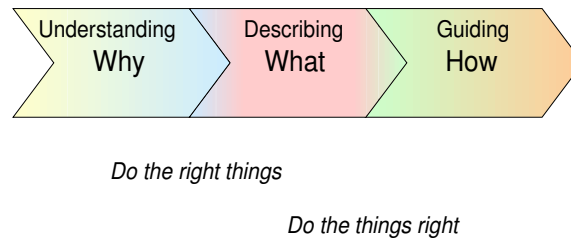


Figure 1: What is Architecture?

Do the right things is addressed in section 2. *Do the things right* is addressed in section 3. The weight of an architecture is discussed in section 4. This structure of the presentation is visualized in figure 2.

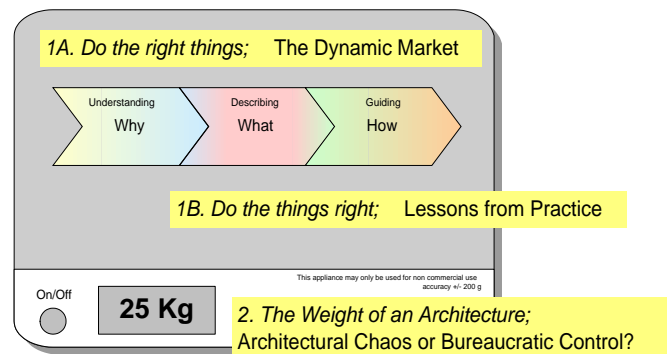


Figure 2: Table of Contents

2 Do the right things; The Dynamic Market

Philips Semiconductors (PS) plays a part in a longer value chain as depicted in figure 3. Typical the components of PS, such as single chip TV's, are used by system integrators, which build CE appliances, such as televisions. These appliances can be distributed via retail channels or via service providers to end consumers.

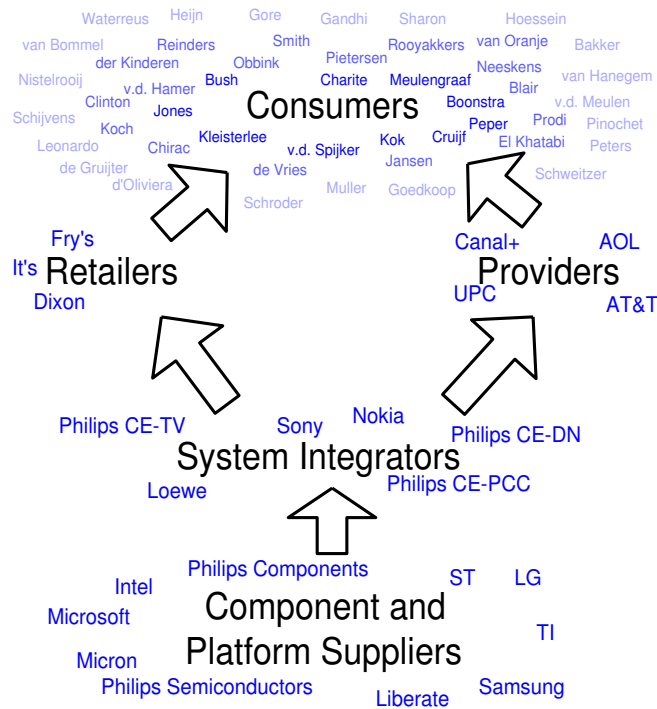


Figure 3: Value chain

One of the major trends in this industry is the magic buzzword *convergence*. Three more or less independent worlds of *computers*, *consumer electronics* and *telecom* are merging, see figure 4; functions from one domain can also be done in the other domain.

The name convergence and the visualisation in figure 4 suggest a more uniform set of products, a simplification. However the opposite is happening. The convergence enables integration of functions, which were separate sofar for technical reasons. The technical capabilities have increased to a level, that required functionality, performance, form factor and environment together determine the products to be made. Figure 5 shows at the left hand side many of today's appliances, in the middle many form factors are shown and the right hand side shows some environments.

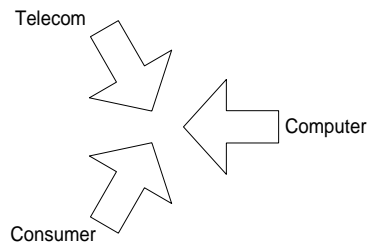


Figure 4: Convergence

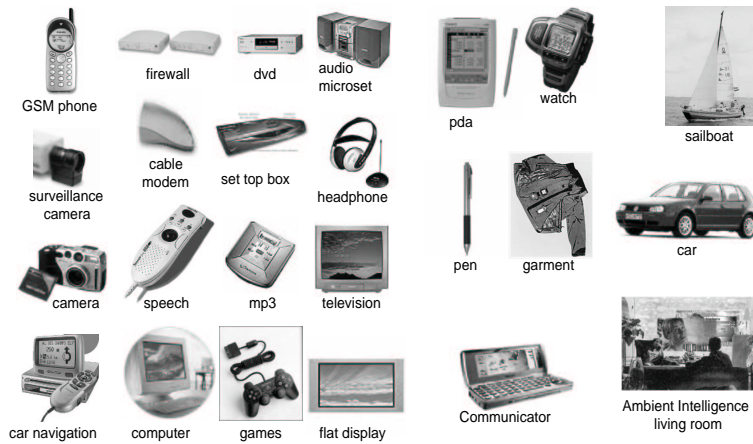


Figure 5: Integration and Diversity

Note that making all kind of combination products, with many different form factors for different environments and different price performance points creates a very large diversity of products!



Figure 6: Uncertainty (Dot.Com effect)

Another market factor to take into account is the uncertainty of all players in the value chain. One of the symptoms of this uncertainty is the strong fluctuation of the stock prices, see figure 6.

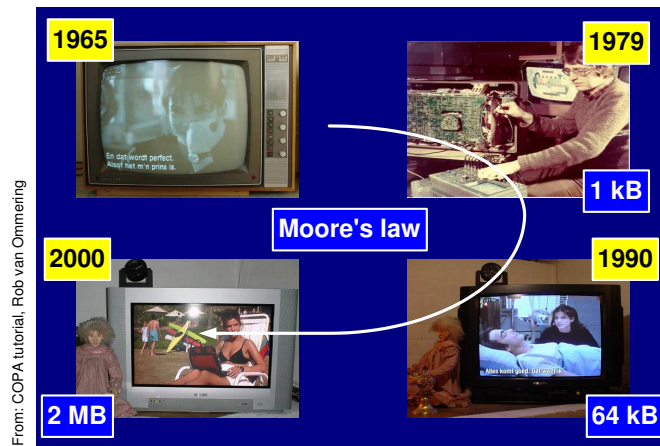


Figure 7: Moore's law

An historical trend is that the amount of software is increasing proportional with Moore's law, see figure 7.

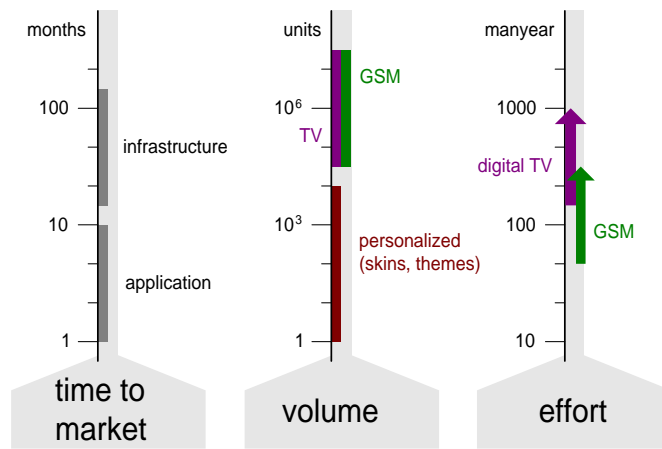


Figure 8: System Integrator Problem Space - Business

From business point of view the products and/or markets of the system integrators can be characterized by *time to market*, *volume*, *effort to create*. In these 3 dimensions a huge dynamic range need to be covered. Infrastructure (for instance the last mile to the home) takes a large amount of time to change, due to economical constraints, while new applications and functions need to be introduced quickly (to follow the fashion or to respond to a new killer application from the competitor). The volume is preferably large from manufacturing point of view (economy of scale), while the consumer wants to personalize, to express his identity or community (which means small scale). As mentioned before the effort to create is increasing exponentially, which means that the effort is changing order of magnitudes over decades. Figure 9 summarizes these characteristics.

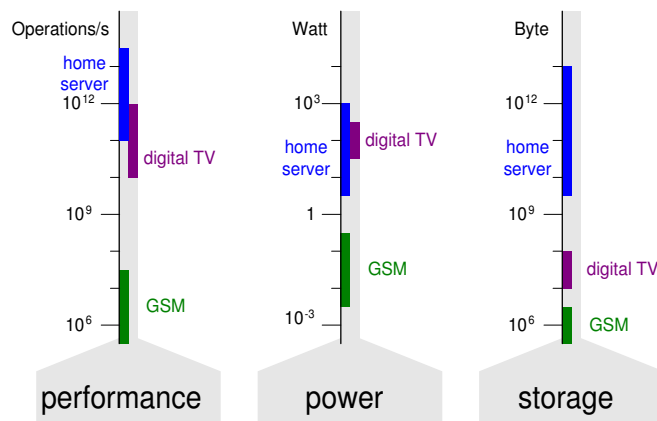


Figure 9: System Integrator Problem Space - Technology

Main technology concerns of the system integrators are *performance*, *power* and *storage*. Again a huge dynamic range need to be covered in these dimensions. Video based applications have much higher processing demands than GSM speech audio. While for power portable appliances like a GSM have severe constraints and should use orders of magnitude less power than TV's or set top boxes. The amount of storage is again highly function dependent, for instance a home server which must be able to store many hours of video needs a huge amount of storage, while the address book of a GSM phone is very limited in its storage needs. The technology parameters and dynamic range are visualized in figure 9.

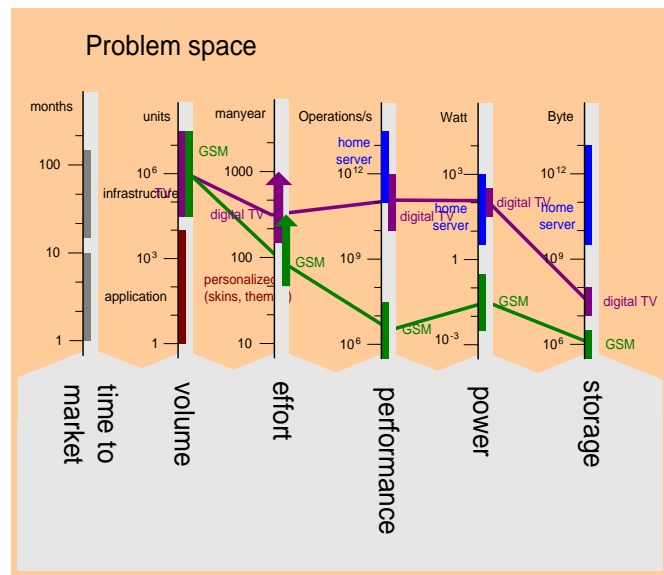


Figure 10: System profile

Combining the figures in one picture enables the visualization of a system profile. Figure 10 shows the profiles for a digital TV and for a GSM cell phone. The profile is not extended to the time to market measure, because several different time constants play a role for both GSM phones and televisions. The device itself, the applications running on the devices, the services offered on device plus infrastructure, and the infrastructure itself all have different time constants.

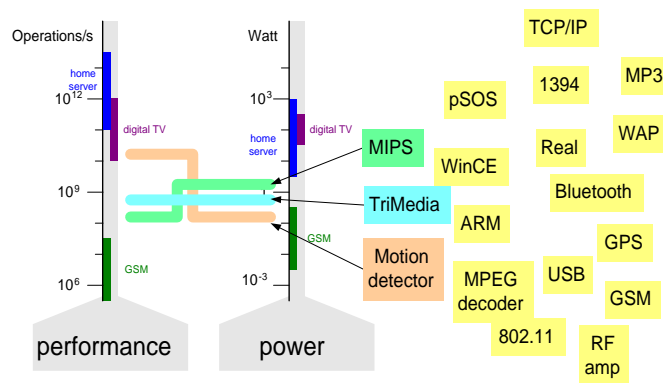


Figure 11: PS Technology solutions

The Philips product division Semiconductors has many hardware and software solutions available in IP-blocks. For a single problem many solutions are available. These solutions differ in their characteristics, such as *performance*, *power* and *storage*. The choice of the solution is driven by the specific product requirements. Figure 11 shows a subset of the available solutions and shows for 3 specific solutions their performance and power characterization.

Systems	Technologies													
	MIPS	TriMedia	MPEG decoder	ARM	Real	GSM	RF amp	Bluetooth	TCP/IP	MP3	pSOS	WinCE	1394	GPS
watch				●	○	○	○	●	○	○	●	○		○
communicator	○	○	○	●	●	●	●	○	●	○	●	○		○
digital TV	●	●	●						○	○	○	○	●	
set top box	●	●	●						○	●	○	○	●	
pda	○	○	○	●	○	○	○	○	●	○		●		○
camcorder	●	●	●			○	○	○	○	○	●		●	○

● required
○ optional

Figure 12: Partial Solution: Configurable Component Platform

The convergence problem (diversity and integration) can be tackled by a platform approach, where all the solutions must be available to be combined in one integrated solution. Figure 12 shows how appliances could be composed from available solutions.

Figure 13 summarizes the exploration of the problem and solution space. The uncertainty and diversity is addressed by *programmability*, *flexibility*, *composable architecture*, *product family approach* and *configurability*. The increased effort

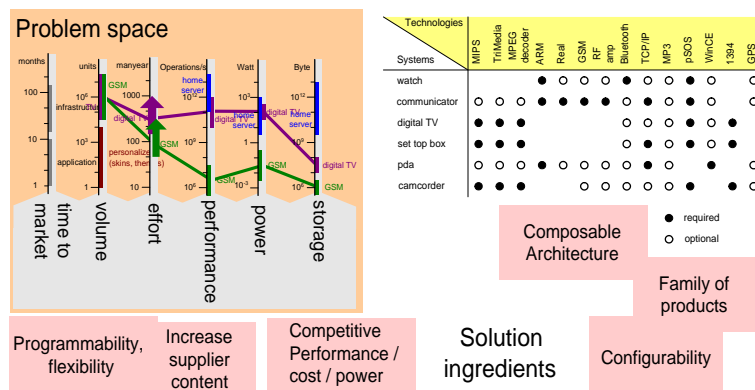
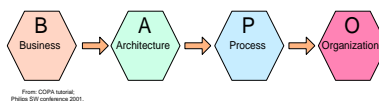


Figure 13: Exploring problem space and solution ingredients

is addressed by *shifting development effort to suppliers*. The dynamic range of requirements is addressed by supplying the right solutions at different *competitive performance/cost/power* points.



Architecture only works if the complementary viewpoints are addressed consistently

Figure 14: More than Architecture

This presentation focuses on architecture. Being good in architecting is not sufficient to be successful in the market. Addressing the Business, Process and Organization (People) issues also is essential for success, see figure 14

Figure 15 summarizes the conclusions of the first part of the article.

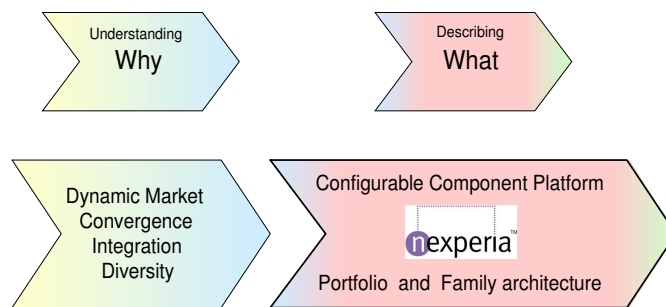


Figure 15: Conclusions Part 1A

3 Do the things right; Lessons from Practice

Creating the solution is a collective effort of many designers and engineers. The architect is mostly guiding the implementation, the actual work is done by the designers and engineers. Guiding the implementation is done by providing guidelines and high level designs for many different viewpoints. Figure 16 shows some of the frequently occurring viewpoints for guiding the implementation. Note that many people think that the major task of the architect is to define **the** decomposition and to define and manage the interfaces of this decomposition. Figure 16 shows that architecting involves many more aspects and especially the integrating concepts are crucial to get working products.

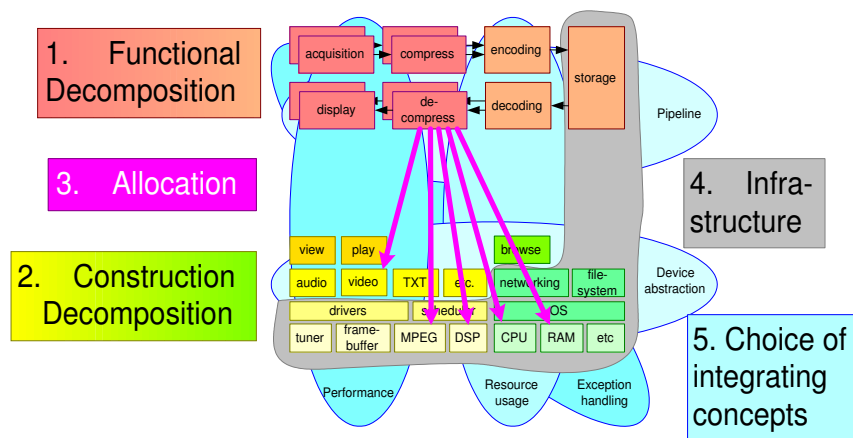


Figure 16: "Guiding How" by providing rules for:

Architecting involves amongst others *analyzing, assessing, balancing, making trade-offs* and *taking decisions*. This is based on architecture information and **facts**, following the needs and addressing the **expectations** of the stakeholders. A lot of the architecting is performed by the architect, which is frequently using **intuition**. As part of the architecting *vision, overview, insight* and *understanding* are created and used.

The strength of a good architect is to do this job in the real world situation, where the **facts, expectations** and intuition sometimes turn out to be false or changed! Figure 17 visualizes this art of architecting.

Many people expect the architect to decompose, as mentioned in the explanation of "guiding how", while integration is severely underestimated, see figure 18. In most development projects the integration is a traumatic experience. It is a challenge for the architect to make a design which enables a smooth integration.

A common pitfall is that managers as well as engineers expect a platform to be stable; once the platform is created only a limited maintenance is needed. Figure 19

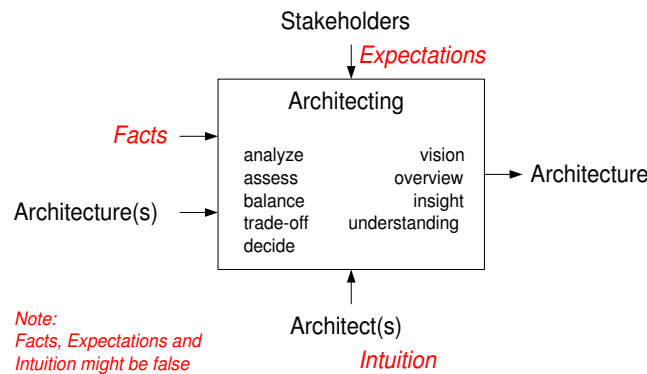


Figure 17: The Art of Architecting

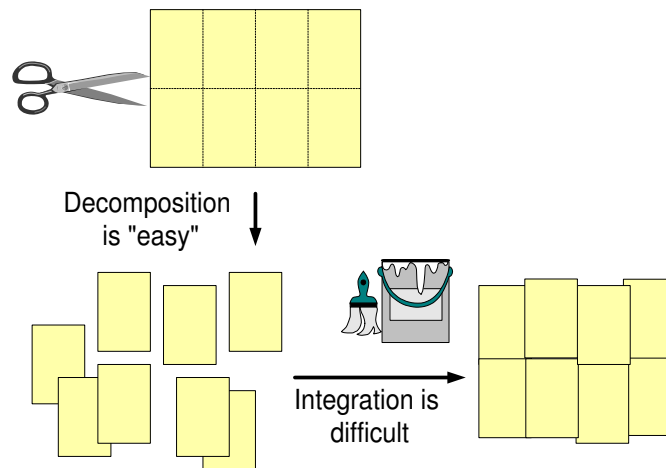


Figure 18: Architecting is much more than Decomposition

explains why this is a myth. A platform is build using technology that itself is changing very fast (Moore’s law again). At the other hand a platform served a dynamic fast changing market, see section 2. In other words it is a miracle if a platform is stable, when both the supplying as well as the consuming side are not stable at all.

The more academical oriented methods propose a ”first time right approach”. This sounds plausible, why waste time on wrong implementations first? The practical problem with this type of approach is that it does only work in very specific circumstances:

- well defined problem
- few people (few background, few misunderstandings)

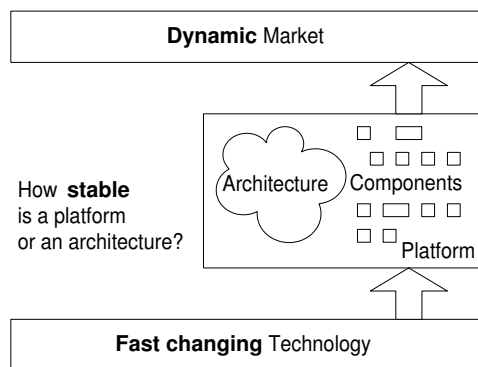


Figure 19: Myth: Platforms are Stable

- appropriate skill set (the so-called "100%" instead of "80/20" oriented people)

The first clause for our type of products is nearly always false, remember the dynamic market. The second clause is in practical cases not met (100+ manyear projects), although it might be validly pointed out that the size of the projects is the cause of many problems. The third clause is very difficult to meet, I do know only a handful of people fitting this category, none of them making out type of products (for instance professors).

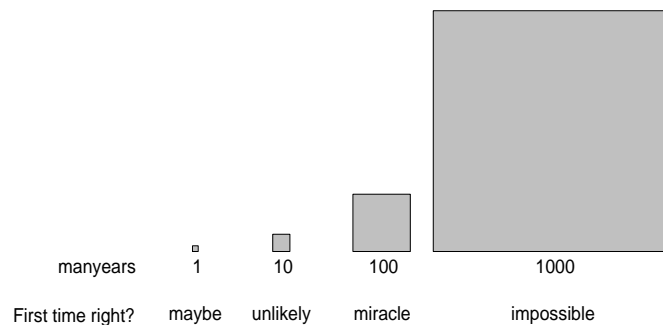


Figure 20: The first time right?

Figure 20 shows the relationship between team size and the chance of successfully following the *first time right* approach.

Understanding of the problem as well as the solution is key to being effective. Learning via feedback is a quick way of building up this understanding. Waterfall methods all suffer from late feedback, see figure 21 for a visualization of the influence of feedback frequency on project elapsed time.

The evolution of a platform is illustrated in figure 22 by showing the change in the Easyvision [2] platform in the period 1991-1996. It is clearly visible that every

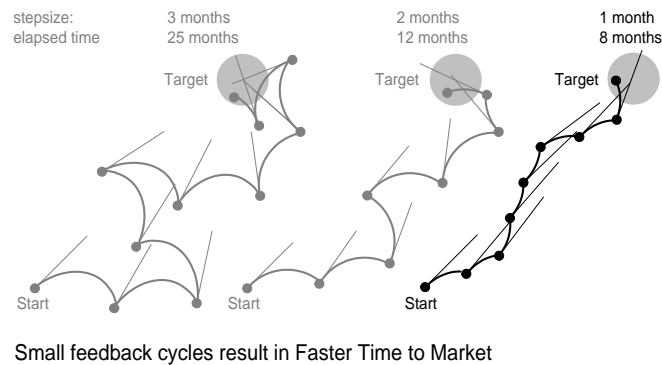


Figure 21: Example with different feedback cycles (3, 2, and 1 months) showing the time to market decrease with shorter feedback cycles

generation doubles the amount of code, while at the same time half of the existing code base is touched by changes.

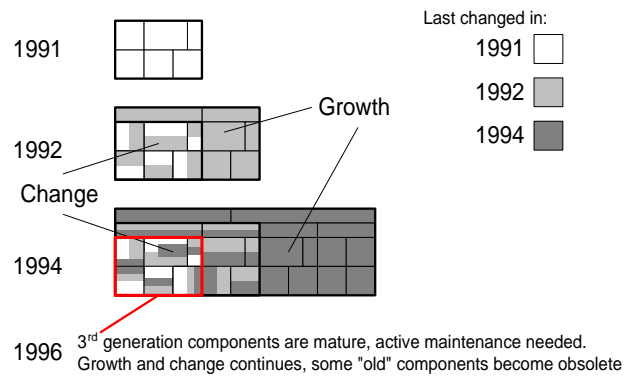


Figure 22: Platform Evolution (Easyvision 1991-1996)

4 The Weight of an Architecture; Architectural Chaos or Bureaucratic Control?

Does an architecture help to be successful in the business or does it harm the success? As always the answer from the architect is: it depends. The crucial success factor is the weight of the architecture.

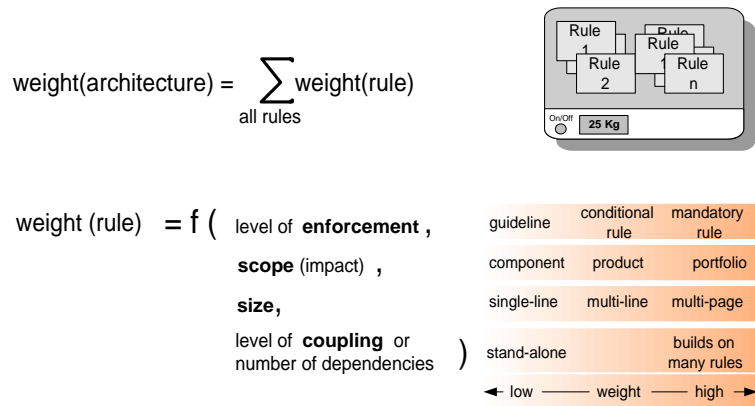


Figure 23: Architecture Weight

Figure 23 gives a definition for the weight of the architecture. The simple definition is that the overall weight of an architecture is the sum of the weight of all rules which together form the architecture.

The weight of a single rule is determined by *level of enforcement, scope (impact), size, level of coupling or number of dependencies*. Figure 23 gives for each of these parameters a scale from low weight to high weight.

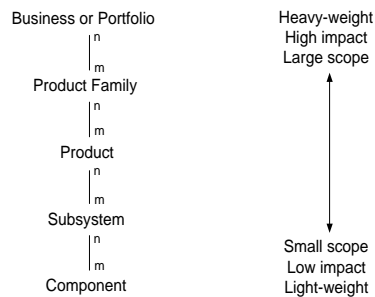


Figure 24: Scope and Impact

Figure 24 zooms in on the scope parameter to make clear the relation between the scope of the rule and the consequence for the weight.

For instance a rule like: *all the functions in all the products of the portfolio*

must (mandatory) return an complete predefined status object as defined in the system design specification, chapter exception handling, making use of the template as described in the coding standards and using the prescribed class and include files as present in the appropriate version of the code repository is a heavy rule (mandatory, portfolio scope, 4 lines of text (without the appropriate references) and depending on other entities (system design spec, template and code standard, prescribed classes and includes, repository).

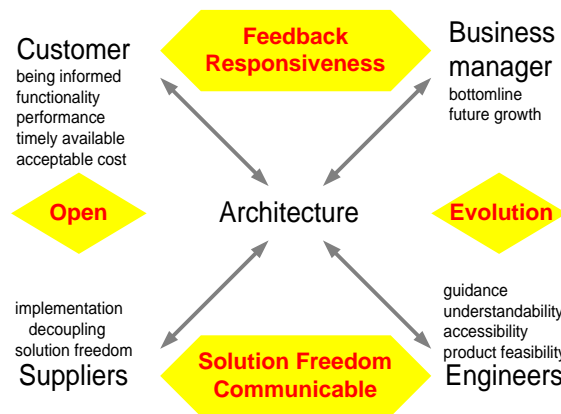


Figure 25: Criteria for an Architecture

So far no judgement is provided for having a *good* or *bad* architecture. In order to make a judgement we need to understand the objectives and concerns of the stakeholders. Figure 25 shows a number of the stakeholders and their main concerns. Note that the stakeholders have different and sometimes conflicting requirements, such is life.

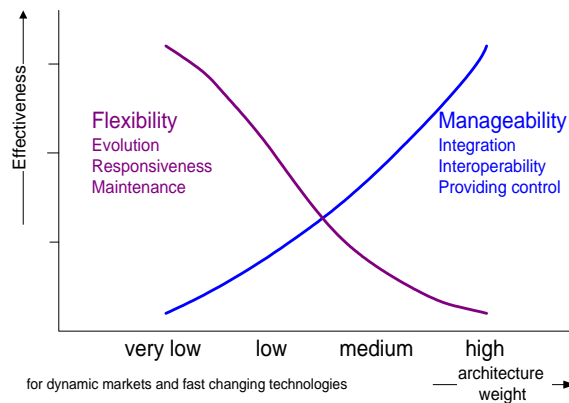


Figure 26: Weight versus Effectiveness

The next step in reaching a judgement is to look at the relation between effec-

tiveness and weight. Figure 26 show this relationship for flexibility (evolution, responsiveness, maintenance) and manageability (integration, interoperability, providing control). Flexibility decreases when the weight increases, while the manageability is proportional with the weight.

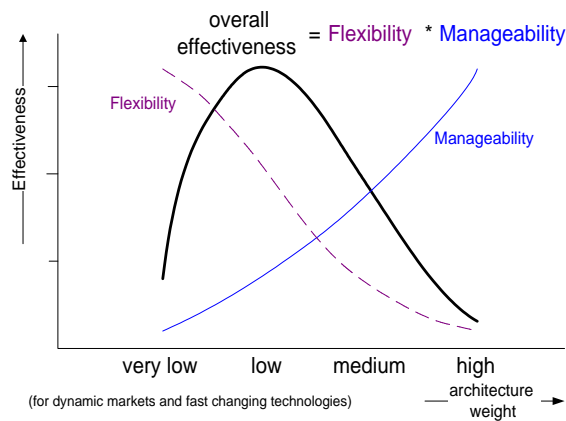


Figure 27: Conclusion Part 2

The question of good or bad depends on the relative importance of flexibility and manageability. For our dynamic markets and fast moving technology flexibility is very important, but for customer satisfaction the manageability is also important. The combination of these 2 requirements is shown in figure 27, where the curve shows that an optimum is achieved when both concerns are sufficiently met.

For different product/market/technology combinations different curves will result. Mature (stable, certain) markets with slow changing technologies, the optimum is determined by manageability and hence a heavy architecture. At the other extreme very dynamic markets and technologies, with forgiving customers or low economic risks will benefit from extreme flexibility and a very light architecture.

5 Light weight how-to

With both the definition of the weight of the architecture and the insight that this weight should be low it becomes possible to look for ways to minimize the weight.

An high impact way of minimizing the weight is to minimize the number of the rules. This can be achieved by providing as much as focus to the architecture as possible. Focus is derived from the market and customer needs. Understanding of the customer helps to understand and therefore to sharpen the requirements, see figure 28.

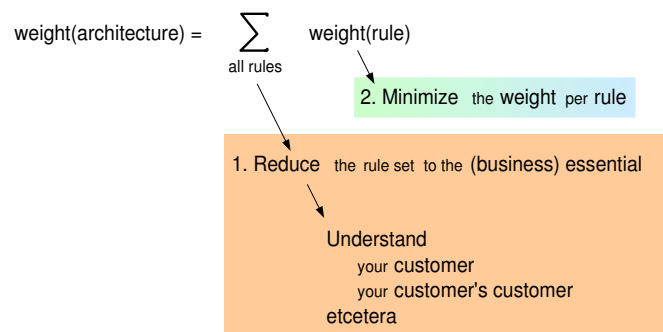


Figure 28: Light Weight How -To

The next step is to minimize the weight of every individual rule. Every parameter influencing the weight of a rule must be minimalized. The *level of enforcement* can be minimized by making as few as possible rules mandatory, work with guidelines as much as possible. The *scope* can be minimized by empowering and delegating as much as possible, in other words let component or subsystem architects make local rules (or better guidelines) for their specific scope. The *size* of a rule is minimized by leaving out details in the rule itself, short conceptual rules are very powerful. The *level of coupling* is minimized by "designing" the architecture rules. Especially multi-view architecting helps to cope with the highly complex reality. One dimensional decompositions result in highly coupled rules to capture aspects from other dimensions.

Figure 29 visualizes the way to minimize the individual weight of a rule.

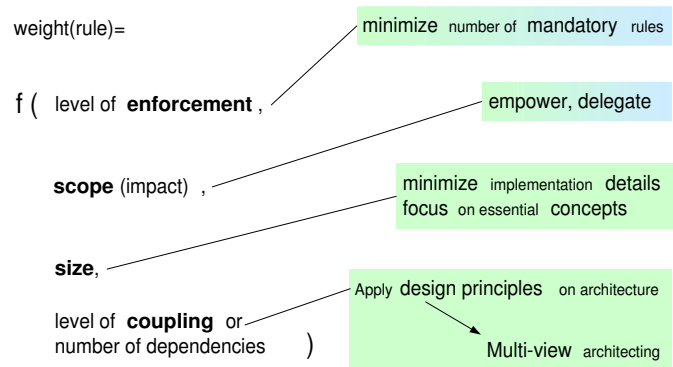


Figure 29: Minimize Rule Weight

6 Summary

Figure 30 summarizes the full presentation. The market to be served is highly dynamic. Lessons from practice show that changes are normal, stability of the solution is the exception. In this dynamic market with a changing solution space the architecture must be light weight.

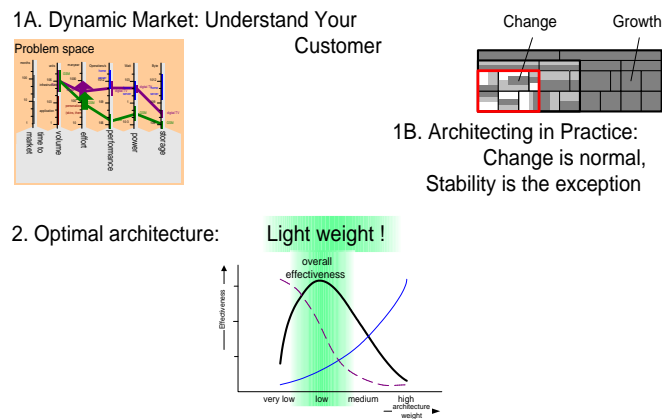


Figure 30: Summary

7 Acknowledgements

This presentation has been enabled by the inspiring and critical comments of:

- Jürgen Müller
- Peter van den Hamer
- Lex Heerink
- William van der Sterren

Remarks and textual improvements were made by Pierre van de Laar.

References

- [1] Gerrit Muller. The system architecture homepage. <http://www.gaudisite.nl/index.html>, 1999.
- [2] Gerrit Muller. Case study: Medical imaging; from toolbox to product to platform. <http://www.gaudisite.nl/MedicalImagingPaper.pdf>, 2000.

History

Version: 2.3, date: June 17,2010 changed by: Gerrit Muller

- minor textual improvement

Version: 2.2, date: June 5, 2008 changed by: Gerrit Muller

- textual improvements
- Explanation added why profile does not extend to time to market

Version: 2.1, date: May 31 2002 changed by: Gerrit Muller

- Updated figure "Criteria for an architecture"
- Updated figure "The art of architecting" and added one sentence to the descriptive text.

Version: 2.0, date: May 29 2002 changed by: Gerrit Muller

- Added the annotated text version