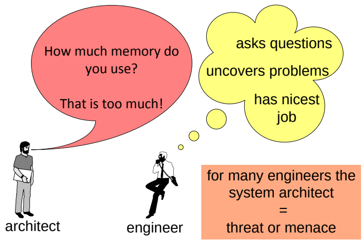
The System Architect; Meddler or Hero?

Annotated presentation given at the DoVo lecture series, November 2001, Eindhoven.


How much memory do you use?
That is too much!
asks questions
uncovers problems
has nicest job
architect
engineer
for many engineers the system architect = threat or menace

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

Abstract

Describes architecting and the task of the architect, with emphasis on bridging the **why**, **what** and **how** of a product. The memory usage of a medical workstation is used as practical illustration.
The introduction of a system architect in an architecture unaware organisation is described. A metamorphosis takes place from a threatening meddler into an appreciated indispensable team member.

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: 3.1 status: finished January 23, 2022

1 Introduction

This lecture is presented at the DoVo[1] lecture series. This lecture series is an ”institute” within Philips Research. Three short lectures (20 minutes) per session are used to share research work between all researchers. This long history also means that many traditions or rituals are followed, such as the opening which positions the speaker in the research line organization. The last section of this article ”CBA” is also tradition.

The presentation itself focuses on a case, which is used to explain the contribution of the architect and the tension this causes in an organisation. Some background material is added about what an architect looks like and about the relation between architecture and research.

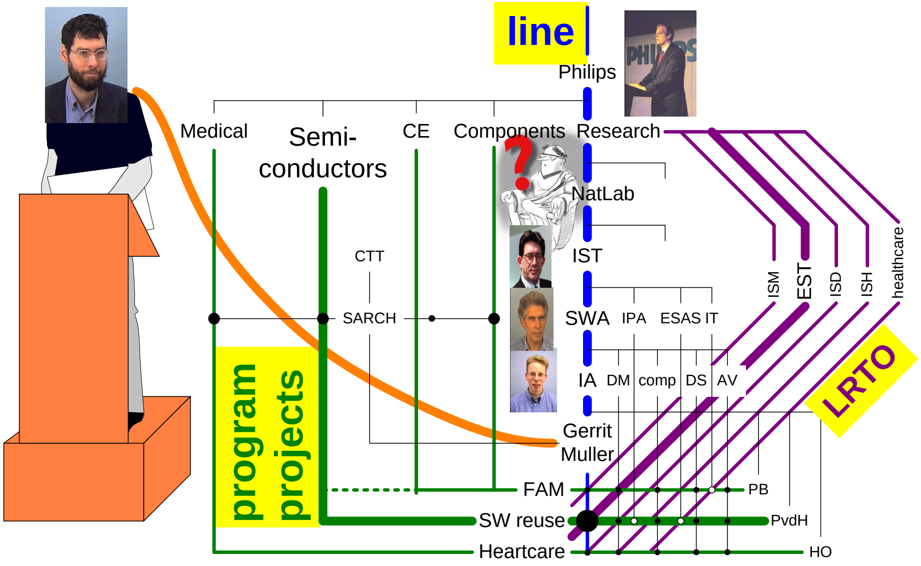


Figure 1: The position within the organisation

Figure 1 shows the position of the speaker in the organisation. However in this, somewhat satiric, diagram other (more) important organisational dimensions are shown:

- program and projects, where a program is result driven
- LRTO (Long Range Technical Objectives), research wide objective driven management
- know how transfer, via the CTT (Centre for Technical Training)

[1] *Donderdag Ochtend Voordrachten*: Thursday morning presentation.

Another interesting tradition is that groups are more often identified by the name of the leader than by the technical competence. This emphasis on the human (leader) contribution is nice.

Recommended literature is the book by Rechtin, ”The Art of Systems Architecting” [6].

2 Practical experience: memory usage in a medical workstation.

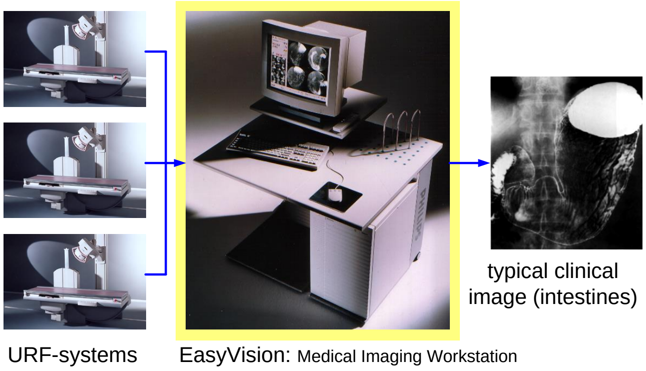

typical clinical image (intestines)
URF-systems
EasyVision: Medical Imaging Workstation

Figure 2: Practical experience; a medical workstation

The medical workstation[5] is an add-on product to existing X-ray systems, introduced in the market in 1992. X-ray systems used to print the imaging results directly on film, by means of a so called CRT-copy, an exact copy of the monitor display on film. The workstation is positioned between X-ray system and printer and adds formatting and layout capabilities. One workstation can serve multiple examination rooms, see figure 2.

The software designers of this product did an excellent job, applying many new technologies in a fruitful way. However many integral design aspects did not get the right attention level, for example memory usage.

Figure 3 shows the memory usage at the beginning of the integration phase. The figure clearly shows a disastrous problem: the system needs much more memory (ca 200 MByte) than available as physical memory (64 MByte). At the bottom of the figure the performance of the system is shown as a function of the memory use. A minor shortage of memory is handled by the virtual memory system, but a major shortage leads to an unacceptable drop in performance.

The architect starts to ask questions about the memory usage, measures it, makes models and budgets. The result is that in cooperation with the software engineers an iterative redesign was implemented. This redesign realized an acceptable memory usage, see figure 4.

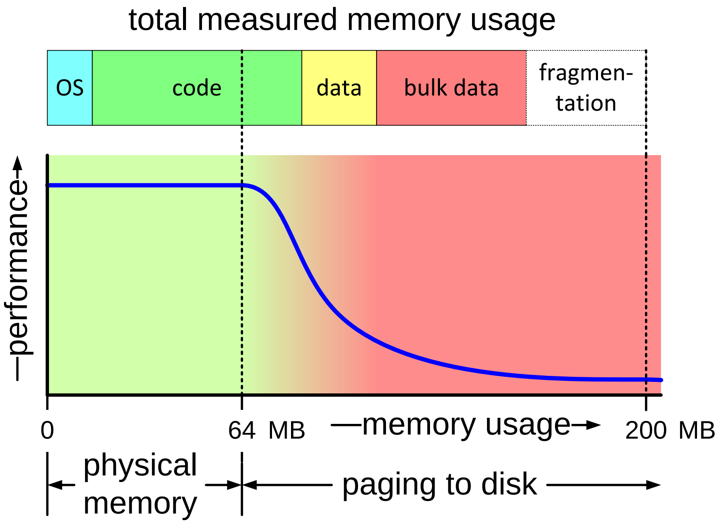


Figure 3: Problem: unlimited memory consumption

Models and budgets are important means of the architect. Figure 5 shows the memory budget, which is based on a process decomposition of the system. This process decomposition enables a manageable granularity of the budget and sufficient implementation freedom for designers. It also enables measurement and verification, because the operating system and the analysis tools use process boundaries as natural resource management boundaries.

The SW engineers of the medical workstation did not experience any performance problem while creating their components, because every individual component fits easily in the available memory. Only when the system is integrated and used under production conditions the performance problem becomes visible. This late visibility and detection of problems is quite normal in the development of complex systems.

Projects run without (visible) problems during the decomposition phases. All components builders are happily designing, making and testing their component. When the integration begins problems become visible. Figure 6 visualizes this process. The invisible problems cause a significant delay[2].

[2] This is also known as the *95% ready syndrome*, when the project members declare to at 95%, then actually more than half of the work still needs to be done.

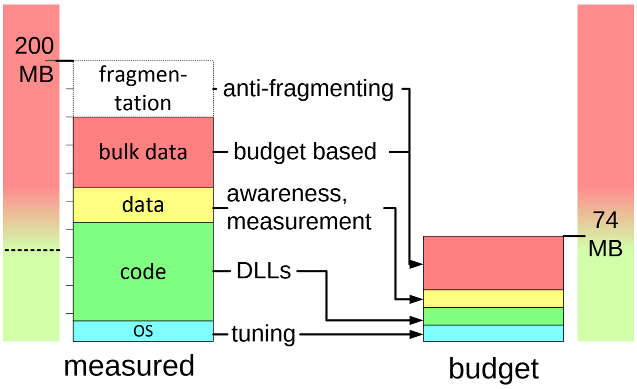


Figure 4: Solution: measure and redesign

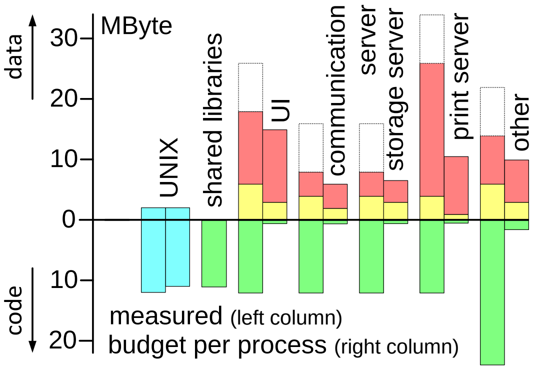


Figure 5: Method: process based budgeting

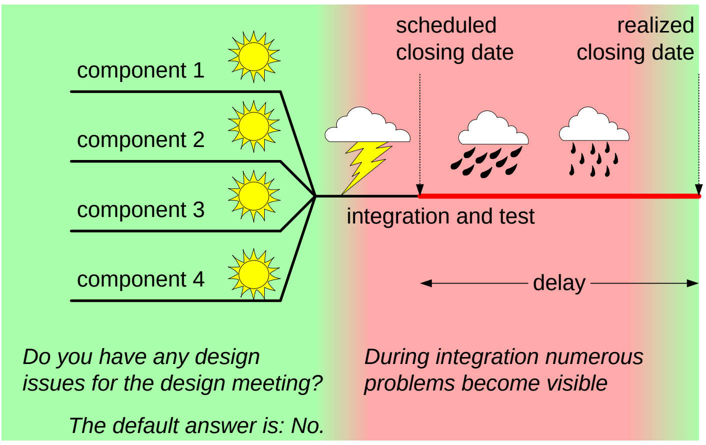


Figure 6: Integration uncovers hidden problems

3 Introducing an Architect

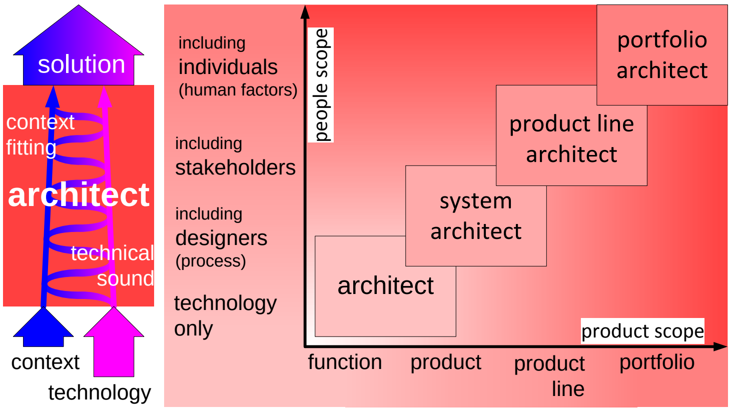


Figure 7: Architecting scope

One of the main causes of the late visibility of problems is the limited context awareness of most component engineers. Most of these engineers are simply not aware of potential problems! This lack of awareness is reflected in the difficulty to plan design meetings. Quite often the engineers don't see the benefit of such a meeting, because they don't see any issues to be discussed.

Many organisations don't have explicit system architects. Sometimes the *best in class* technical specialist gets the architect title imposed. In both cases the introduction of a broad system architect causes a shock effect in the organisation.

Figure 7 shows that the scope of architects widely varies. The common denominator for all these architects is the bridge function between context and technology (or problem and solution). An architect needs sufficient know-how to understand the context as well as the technology, in order to design a solution, which fits in the context and is technical sound at the same time.

In general increasing the product scope of an architect coincides with an increase in people scope at the same time.

Figure 8 shows the phases an organisation is going through in a typical project where an architect is introduced.

As long as individual designers can work independently the collective mood is great, while an architect is mostly perceived as a threat or a menace, see also figure 9. As soon as the integration starts all invisible problems suddenly become visible, the beginning of a crisis, which changes the mood to poor.

An architect who proves himself in this difficult stage, by hard work, brainstorming, trouble shooting and problem solving earns a lot of credit. This changes the appreciation of the architect dramatically, suddenly he becomes an indispensable team member! After completion of the integration the mood returns to good. In a

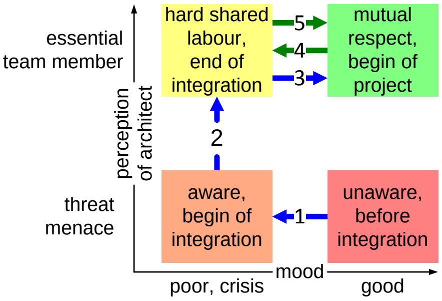

essential
team member
threat
menace
perception
of architect
hard shared labour, end of integration
mutual respect, begin of project
aware, begin of integration
unaware, before integration
5
4
3
2
1
poor, crisis
mood
good

Figure 8: Architecture awareness evolution

next project, again during integration the mood will degrade, even excellent architects will not prevent this. However this crisis will be less severe.

An architect always starts to ask questions, to build up understanding and overview. While sampling the problem and solution domain in this way, he always discovers some weak spots. The identification of problems and risks is often based on a judgement or an opinion.

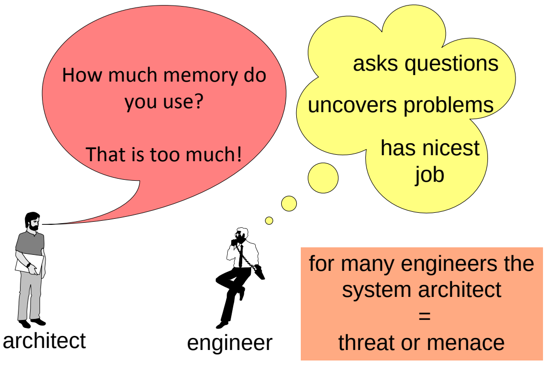

How much memory do you use?
That is too much!
asks questions
uncovers problems
has nicest job
for many engineers the system architect = threat or menace
architect
engineer

Figure 9: The engineer's perception of the architect

The judgement or the opinion is based on a sample of all data, fitting in the limited available time. Despite the incompleteness of the data and despite a lack of domain and solution know-how the architect forms an opinion anyway. The architect does the top level design, the nice hand-waving work, without being too concerned with the nasty details.

Whenever you want to benefit from the architect's expertise, by asking for a solution, the architect only worsens the problem, by showing even more hidden problems.

4 What is architecting?

Architecting in product creation spans from *understanding* the **why**, via *describing* the **what** to *guiding* the **how**, as shown in figure 10. Or in even more popular terms: *do the right things* and *do the things right*

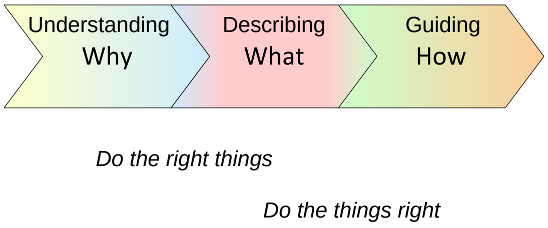


Figure 10: Architecting visualized

Architecting is a job which is done by all members of the product creation team, however the architect is responsible for the consistency and balance of **why**, **what** and **how**

A useful top level decomposition of an architecture is provided by the so-called "CAFCR" model, as shown in figure 11. The *customer objectives* view and the *application* view provide the **why** from the customer. The *functional* view describes the **what** of the product, which includes (despite the name) also the *non functional* requirements. The **how** of the product is described in the *conceptual* and *realization* view, where the conceptual view is changing less in time than the fast changing realization (Moore's law!).

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

Figure 12 shows these 5 views with some relevant issues with respect to the illustrated memory usage. The customer objectives are expressed in a number of keydrivers, constrained by a street price of 50k$.

The application and functional view are here shown together, expressed by a typical case with 3 connected X-ray system, where an examination has a typical size of 20 images, which are auto-printed on 3 film sheets.

The conceptual view contains a decomposition, amongst others in import, database and print servers. It also contains concepts to constrain the memory usage, such as anti-fragmentation and dynamic link libraries.

The realization view contains the actual measured memory usage as well as the budgeted usage.

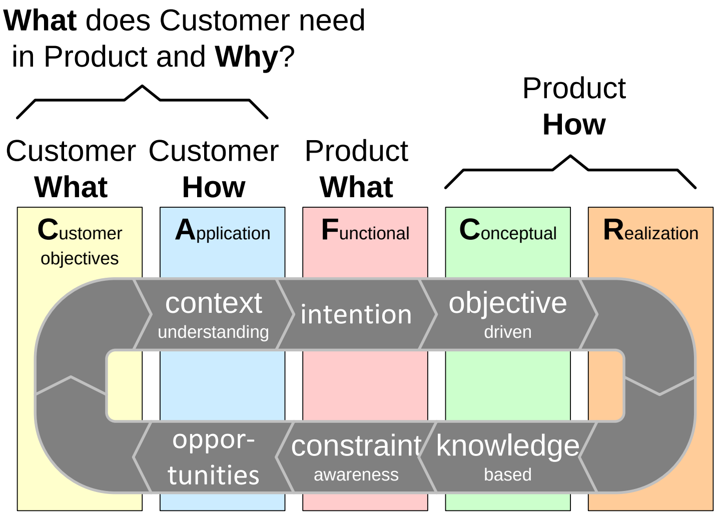


Figure 11: Five viewpoints for an architecture. The task of the architect is to integrate all these viewpoints, in order to get a *valuable*, *usable* and *feasible* product.

The **how** of the product is created by many specialists. The **how** is guided by the architecture. At least 5 views are required for guidance:

- functional decomposition
- construction decomposition
- allocation of functions to construction elements
- infrastructure
- integrating concepts

Figure 13 visualizes these 5 **how** views.

Figure 14 and 15 shows a question generators, which can be used to uncover potential problems. This question generator is based on a discrete 3-dimensional space, where every point in this space can be used to formulate a question. The axis of this space are:

- functions
- (HW) components
- characteristics

The question in any point in space looks like:
”How about *characteristic c* in *HW component h* when performing *function f*?”

Nearly all questions formulated in this way will get an answer *unknown*, which in most cases means *here is a potential problem*.

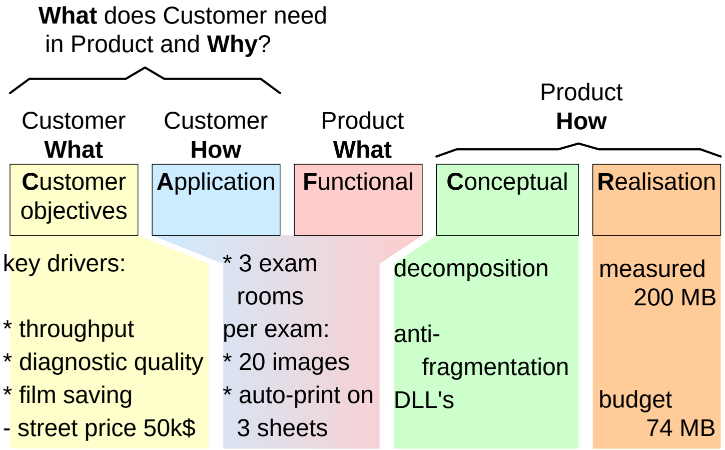


Figure 12: The customer issues which are relevant for the case illustrated in this article

Note that a good architect uses a more refined question generator, which uses a priori know-how to select the really relevant questions. The a priori know-how includes:

- importance, value of functions and performance
- how critical, sensitive technical solutions are
- people and organisation bias, strengths and weaknesses

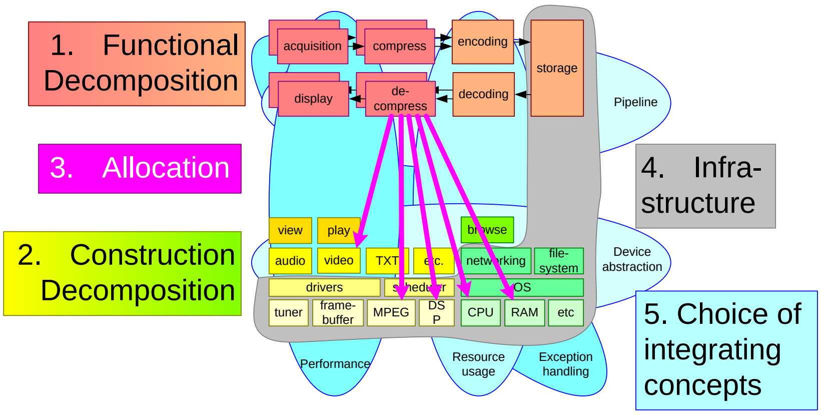


Figure 13: Guiding **how** by providing five *how*-viewpoints

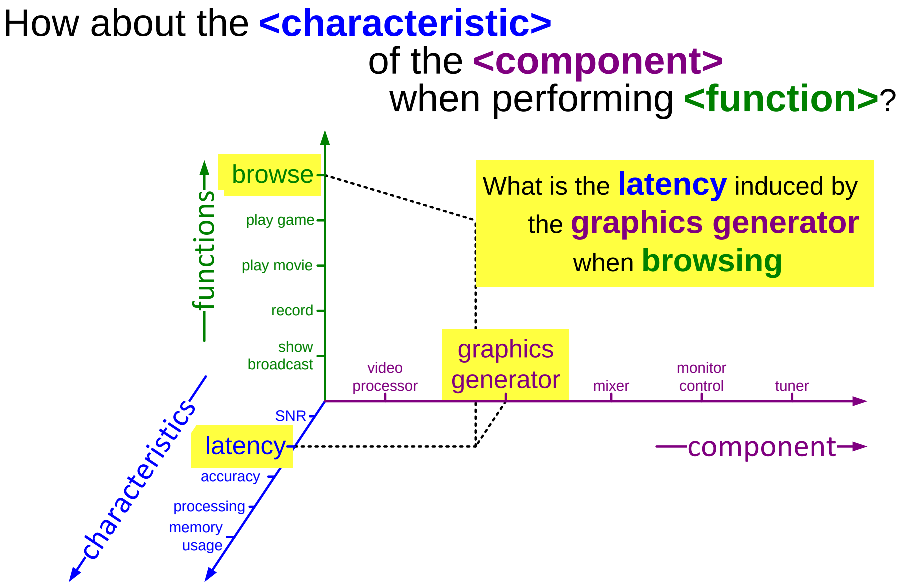
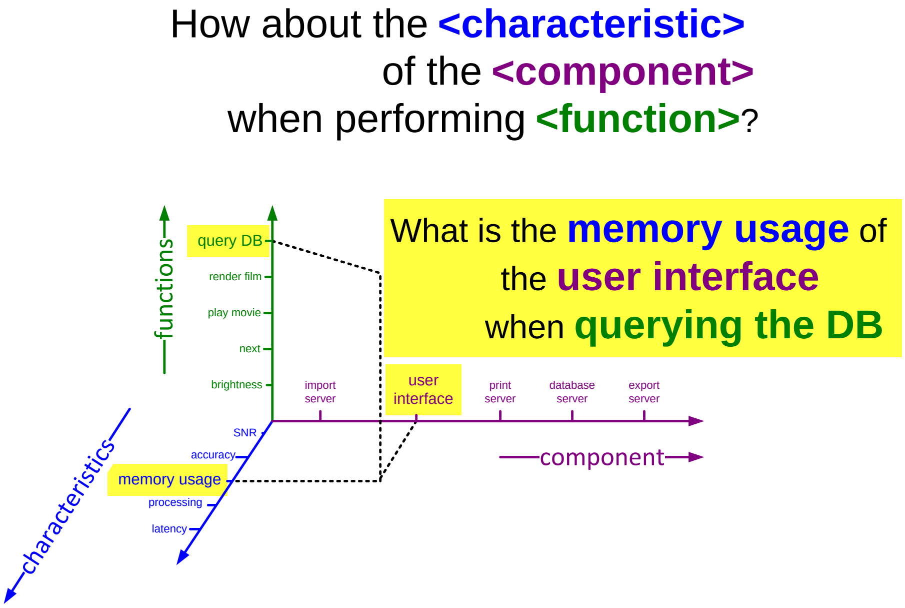


Figure 14: Question generator



How about the <characteristic> of the <component> when performing <function>?

What is the memory usage of the user interface when querying the DB

Figure 15: Question generator

5 CBA: Conclusions, Benchmarking and Acknowledgements

5.1 Conclusions

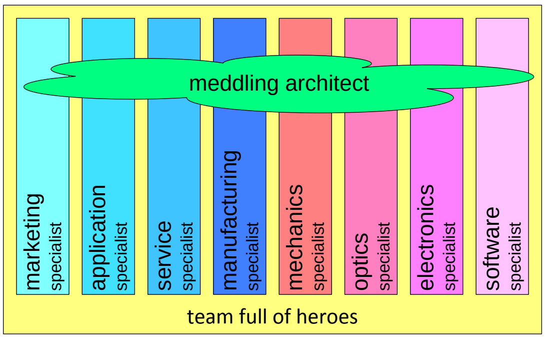


Figure 16: The meddling architect as complementing factor in a team full of heroes

The work of the architect is always overlapping with the work of others, see figure 16. The integration of views is the main added value of the architect. Sometimes the architect is meddling in the work area of specialists, with the intention to serve the overall objective. The architect can only be succesfull by virtue of the rest of the team, so the architect is not the hero, all team members are heroes.

5.2 Benchmarking

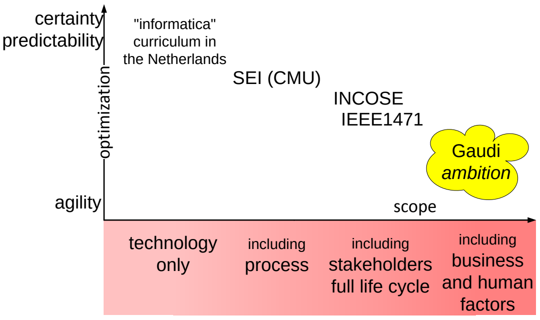

certainty
predictability
optimization
agility
"informatica" curriculum in the Netherlands
SEI (CMU)
INCOSE
IEEE1471
Gaudi ambition
scope
technology only
including process
including stakeholders full life cycle
including business and human factors

Figure 17: Positioning the Gaudí research ambition in the worldwide efforts

Most software engineering oriented institutes in the world, such as SEI(CMU) stress the importance of systematic approaches and the use of processes. This emphasis often results in formalization, which endangers the flexibility and adaptivity to fast changing technology and markets[3]

For wider scopes formalization is more difficult and less fruitful. Insight, understanding and overview are more important in a broad perspective. Figure 17 positions several worldwide activities with respect to formalization level and scope.

The System Engineering community, which is mostly flourishing in the military and aerospace industry, is very mature with respect to requirements engineering, stakeholders and life cycle management. Part of that work is consolidated in ”standards” for best practices, such as IEEE1471.

The ambition of the Gaudí project is to provide more insight in the *art* part of architecting, which quite often is related to human aspects.

5.3 Acknowledgements

Figure 18 shows some of the participants of the SARCH courses. The participants of these courses provide me always with valuable feedback and often trigger new insights.

Hammer, Dieter
Hoogenstraaten, Wil
Mueller, Juergen
Gieles, Hans
Eggenhuisen, Huib
Kloprogge, Raymond
Engel, Bas
van Rijnsoever, Bart
Driesen, JGH
Schelkers, Raymond
van der Heijden, Jaap
Vermeulen, Gerry
Stroucken, Marc
Wijnstra, Jan Gerben
Algra, Egbert
Derks, Frans
Faber, Albert
Aarts, Peter
Watabe, Yasuma
van Ouwerkerk, H
Huijnen, Ton
Gonot, Mathieu
van Splunter, Andre
van Rooijen, Joost
Verberkt, Mark
van der Linden, Wim
Patrzalek, Jarek
Vergoossen, Theo

Gijsbers, Rob
Huis in 't Veld, Robert
Joosten, Jan
Mulder, Alwin
de Wit, Paul
Poesse, Jan
Spaak, Wim
Thus, Frank
van Velden, Jeroen
van Venrooy, Roland
Dobbelsteen, Jan
de Waal, Klaas
Muijen, M
Peters, Jo
van Bommel, Pieter Jan
Thijssen, Henk
Boot, Gert Jan
Vullings, Erik
Vermeer, Ad
Peeters, Bob
Obbink, Henk
Bas, Han
Rankers, Adrian
Akiwumi-Assani, Olu
Gopalan, Rajaraman
Misdom, Han
Schatorie, John
Boer, Richard

Penners, Maurice
America, Pierre
Jaspers, Peter
Versteijlen, Joost
Beelen, Peter
Blijd, Jarl
Dijkema, Marcel
Roelandt, Werner
Janson, Paul
Bandakka, Mahesh
Ledeboer, Jodie
Geron, Nic
Zieringer, Peter
Beuk, Leo
Koolen, Gertjan
Koushik, Sudeendra
Milosevski, Vlatko
van den Broek, Ger
de Kruif, Peter
Daenen, Steven
Soepenberg, Gerben
Bingley, Peter
Follon, Roel
Elzinga, Onno
van den Donker, Piet
Zwaans, Ben
Harmsze, Francoise
Jansen, Tom

van Gogh, Clemy
Wissink, Getty
Engelsma, Erwin
Stut, Wim
Luttikhuizen, Paul
Bruin, Jan
Gooren, Huub
den Dekker, Wim
van der Laak, Eric
Crins, Wim
Heerink, Lex
Schippers, Alef
Schreppers, Jurgen
Deckers, Robert
van Balen, Auke
Huiban, Cristian
van Loon, Gerard
van den Heuvel, Patrick
Lobo, Lincoln
van de Meulenhof, Dennis
Houtepen, Rob
Hofsink, Robert
Buurman, Hans
Zondag, Eddy
Veldmans, Ferdinand
Merkus, Paul
van Tuijl, Frank
Wouters, Kees

van der Sterren, William
Soede, Michiel
van Bommel, Luc
Krikhaar, Rene
van den Brink, Johan
Ham, Kees
Bos, Erik
Pijpers, Frank
Medema, Jeroen
Kaag, Bjorn
Giesselman, Timo
Vos, Frans
de Greef, Pierre
Fischer, Stefan
Pu, Xuemei
Boom, Sjirk
ten Pierick, Henk
Stroucken, Louis
Young Tai Liu
van der Steen, Marcel
Siereveld, Ad
van Bakel, Gerian
Engbers, Rene
van Wetten, Frank
Stevers, Frank
Wubben, Rob
Schellingerhout, Nico
Vugts, John

Figure 18: A small subset of contributors, here mostly participants of the System Architecture course

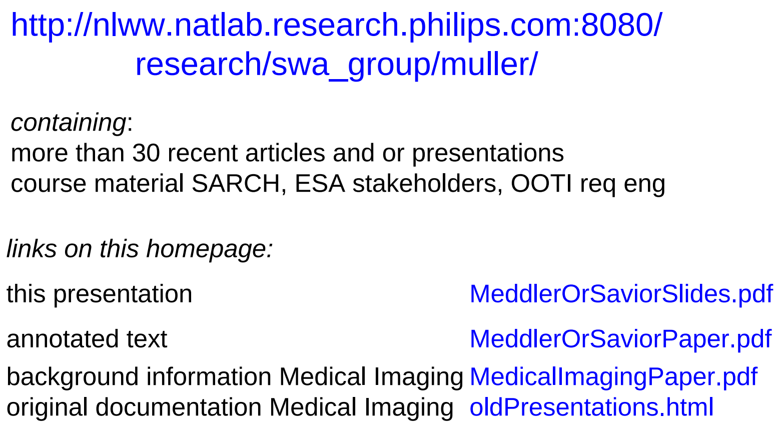
Jürgen Müller again critically reviewed the presentation, helped to streamline it and discussed the right sequence of presenting. William van der Sterren and Peter

[3] Strictly speaking formalization and agility are not contradictions. Agile formalization can be supportive. The actual danger is in the less skilled people applying the systematic approaches in rigid ways.

van den Hamer suggested a lot of improvements. Jaap van der Heijden pointed out the areas of interest for the target audience.

Pierre America, Jaap van der Heijden, Niek Lambert and Milan van den Muyzenberg patiently listened to the trial run and politely pointed out the missing steps and unclear issues.

5.4 Gaudí homepage



http://nlww.natlab.research.philips.com:8080/
research/swa_group/muller/

containing:
more than 30 recent articles and or presentations
course material SARCH, ESA stakeholders, OOTI req eng

links on this homepage:

this presentation	MeddlerOrSaviorSlides.pdf
annotated text	MeddlerOrSaviorPaper.pdf
background information Medical Imaging	MedicalImagingPaper.pdf
original documentation Medical Imaging	oldPresentations.html

Figure 19: Gaudí homepage

Figure 19 shows the URL for the Gaudí homepage, see[4] for the www internet URL.

See also the bibliography for more recommended reading.

6 Does architecture belong in the research laboratory?

Research laboratories are used to in-depth technology research, where system know how is mostly required to build a proof of concept. Architecting at the other hand is very broad and not tangible. Figure 20 shows a model for technology management[2], based upon 3 phase cycle.

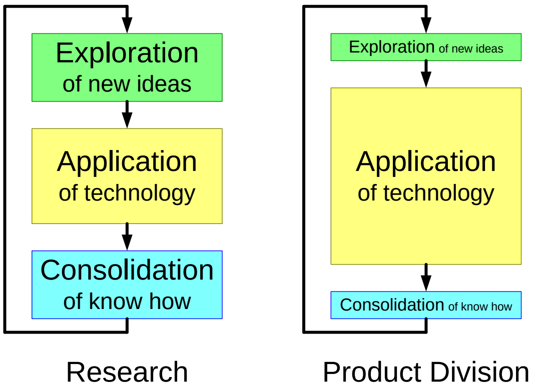


Figure 20: Architecture and research? The technology management cycle

One of the main functions of research is technology exploration, where the technology application is needed for learning and proof of concept. The consolidation is needed for the transfer.

Product divisions focus on short and medium term business objectives, research is taking care of the long term. Redefinition of architecting as enabling technology to specify, design and integrate complex systems, fits architecting in a natural way in this technology management model. It makes quite a lot of sense to explore architecting methods as well as consolidate architecting methods.

However the application of architecting methods often needs the full product creation context, which means that the application often happens in close cooperation with the industry, the so called **industry as laboratory**.

The next question is: *Is architecting research* **scientific***?*. The answer of this question requires a philosophical basis, what do we mean with *scientific*? Figure 21 shows several sciences as a spectrum with respect to the hardness level of the scientific methods. This figure shows that architecting methods span a significant range of scientific methods. Or in other words the architecture researcher borrows methods from other scientific paradigms, striving for as hard results as possible[4]

[4] Which means that the results can be very soft. How to distinguish plausible results from an integer researcher from convincing results from a charlatan?

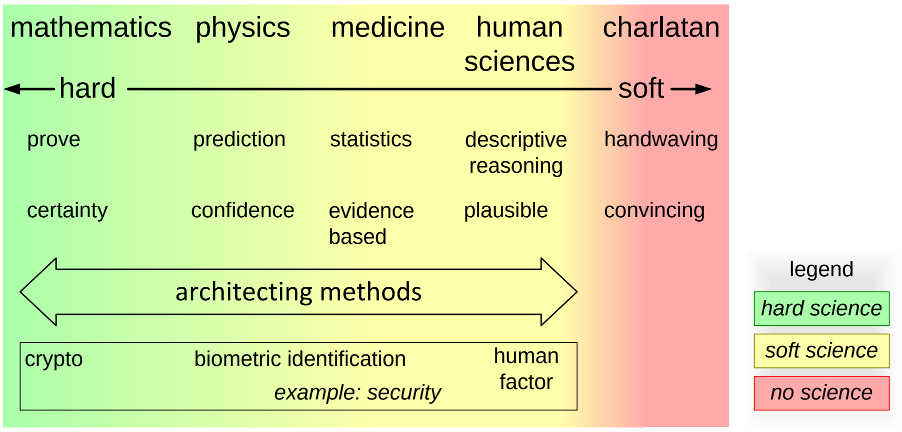


Figure 21: Is architecting scientific?

7 What is an architect?

The architect is good technical educated engineer, who has grown into an architect, see [3]. The main growth direction are:

- technical generalist
- business insight
- process insight
- human insight

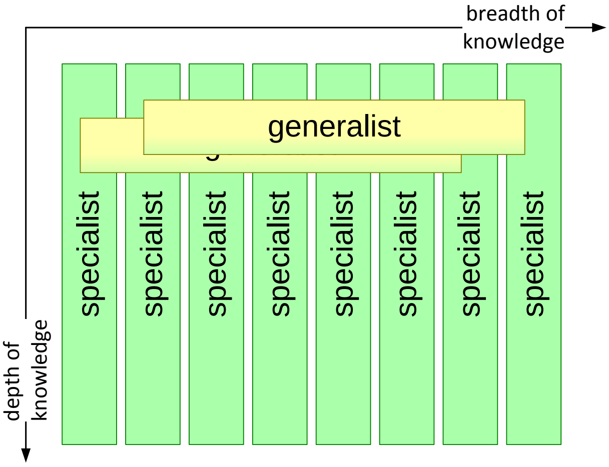


Figure 22: The architect as integrator

The role of the architect is an integrator role, as shown in figure 22, which is highly complementary to the specialists.

The real world is less black and white, a complete spectrum exists between specialists and generalists, as shown in figure 23. Architects must have sufficient roots in the technical domain, which means that they must experience angineering in at least one discipline. Later when growing in the above mentioned directions the difficult challenge is to maintain sufficient technical know how and feeling.

References

[1] Frederick P. Brooks. *The Mythical Man-Month*. Addison Wesley, 1975, ca. 1995.

[2] Hay Management Consultants. Technology management cycle. Hay Managament Consultants showed me this model in 1997/1998, taken from

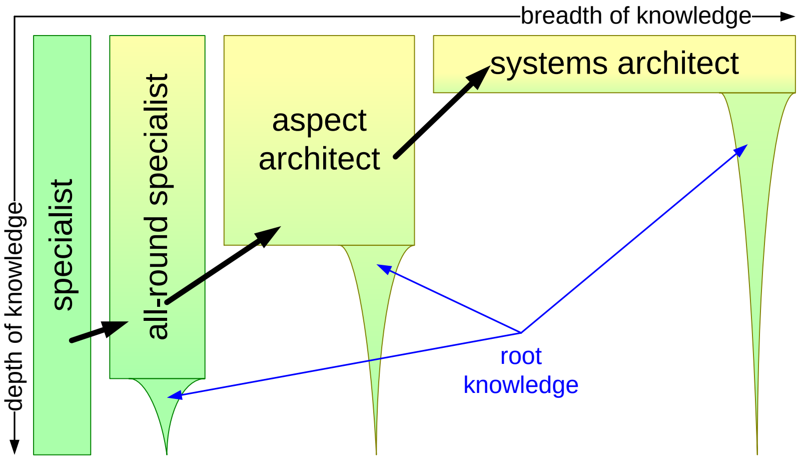


Figure 23: The architect maintains technical roots

an article by a Japanese author. The original title of the Japanese article is unknown.

[3] Gerrit Muller. The arisal of the system architect. `http://www.gaudisite.nl/MaturityPaper.pdf`, 1999.

[4] Gerrit Muller. The system architecture homepage. `http://www.gaudisite.nl/index.html`, 1999.

[5] Gerrit Muller. Case study: Medical imaging; from toolbox to product to platform. `http://www.gaudisite.nl/MedicalImagingPaper.pdf`, 2000.

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

History

Version: 3.0, date: November 1, 2001 changed by: Gerrit Muller

- renamed organisation sheet
- added who is gerrit sheet, what is Gaudí sheet
- replaced Medical Workstation sheet
- redesigned problem statement sheet

- annotated the detailed memory budget
- added architecting scope sheet
- removed "what is architecting", "guiding how"
- replaced appreciation curve by awareness path
- replaced engineers perception sheet with more visual picture
- replaced the integrating CAFCR with more specific CAFCR sheet
- replaced digital TV example with Easyvion in Question Generator
- redesigned conclusion
- benchmarking: color annotation added and changed vertical axis
- changed order of FAQ sheets
- changed logo
- appended all removed sheets to end of presentation: ”sheets that didn’t make it into the presentation”

Version: 2.1, date: October 22, 2001 changed by: Gerrit Muller

- more conclusive conclusion
- moved architect characteristics to "What is architecting"

Version: 2.0, date: October 15, 2001 changed by: Gerrit Muller

- Title changed: Hero instead of Savior

Version: 1.2, date: October 15, 2001 changed by: Gerrit Muller

- improved readability:
    - Benchmarking
- memory budget slides redesigned

Version: 1.1, date: October 12, 2001 changed by: Gerrit Muller

- improved readability:
    - Integrating CAFCR
    - "Question generator"
    - "Is architecting scientific?"
    - "Technology management cycle"
- memory budget slides redesigned

Version: 1.0, date: October 10, 2001 changed by: Gerrit Muller

- added a practical experience, 4 slides on memory use in a medical workstation
- changed the sequence, work more from practical case ”upward”
- added a section ”Does architecture belong in the research laboratory?”

Version: 0.1, date: October 8, 2001 changed by: Gerrit Muller

- Slide 7 title changed
- Slide 8 (Question Generator) changed

Version: 0, date: October 3, 2001 changed by: Gerrit Muller

- Created, no changelog yet