Coupling Enterprise and Technology by a Compact and Specific Architecture Overview

Gerrit Muller
Embedded Systems Institute
Gerrit.muller@embeddedsystems.nl

Abstract. Enterprise level projects tend to be delayed and to exceed budgets. Original expectations and targets are not met by these projects. Many solutions have been proposed to address this problem, one of them the application of Systems Engineering at large. In this paper we show how a compact *Architecture Overview* is a helpful instrument to keep enterprise level projects on track.

Practical experience shows that such an overview provides a quick insight in project progress and in potential threats. Principal customers and managers should ask the architects for this overview. Benefit for the architects of making the overview is that it helps to position their work in the enterprise context. Experience also shows that pragmatic system engineering practices, such as process orientation and project management are complementary to the proposed overview. This combination of overview and pragmatic systems engineering is very powerful.

One of the main issues in creating an architecture overview is the need for *breadth*, what needs to be included and for whom, and the balancing act of providing sufficient *depth*, what are crucial details that are part of this top-level description. Also the way of describing is discussed, from stakeholder needs to ambiguity and the level of formalism.

Introduction

A broad architecture overview relating specific facts to specification and design choices is a powerful tool in product creation. The architecture overview provides:

- Focus on customer and business
- Direction and guidance to the project team
- Insight in important choices and risks
- Attention for issues that go beyond organizational entities

This article has been inspired by the *Dutch Championship ICT architecture* that has been organized yearly since 2004 (see also lessons learned). Participants of the championship come from many different domains: from data warehousing for supermarkets or publishers to embedded systems such as electron microscopes. In 2006 many government owned projects participated. The experiences of the jury, 9 people representing a broad set of architecture and management views, have been captured in this paper. This experience is transformed into a stepwise approach with recommendations to facilitate practical deployment.

Approach

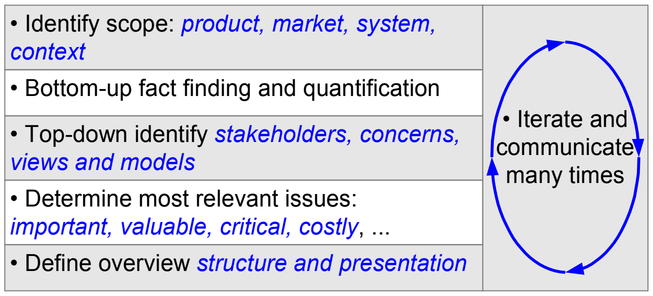
• Identify scope: product, market, system, context
• Bottom-up fact finding and quantification
• Top-down identify stakeholders, concerns, views and models
• Determine most relevant issues: important, valuable, critical, costly, ...
• Define overview structure and presentation
• Iterate and communicate many times

Figure 1. Approach to Create Architecture Overview

The benefits mentioned in the introduction require a compact and to-the-point overview. We present in this article a stepwise approach to create such an overview, see Figure 1. We recommend using time-boxes for the steps of this approach. For the first iterations we recommend 15 to 60 minutes per time-box, for later iterations we recommend hours to maximum a few days per time-box.

The architect leads this process and captures the result in a compact and accessible document. The architect needs to have an independent position relative to the stakeholders, and the often silo-oriented organizations. The explicit and compact format should facilitate communication across organizational walls.

The first step is to define the scope of the architecture overview. The next step is bottom-up by exploring facts. We strongly recommend quantifying facts as much as possible, to ensure that this exploration is sufficiently specific. Next step is to work top-down, where we follow IEEE 1471: Who are the stakeholders, what are their concerns, what are relevant views, what models are needed to describe these views? The word *relevant* is more extensively addressed in the next step, where we need to identify the most valuable and most important issues from customer perspective and the most critical and expensive issues from design perspective. This selection step is the most difficult and the most crucial step in the entire approach. The structure and the way to present the overview can now be determined. All the mentioned steps are repeated several times. This iteration is used to incorporate insights obtained in earlier passes. For instance, the scope may move or get sharper based on new stakeholder insights.

Fact Finding and Quantification

In Figure 2 we show an extended CAFCR model (Muller 2004) as a framework for fact-finding and quantification. The CAFCR model is a top-level decomposition of an architecture. 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 extension provided here is the *internal Operational view* describing all the company internal considerations, ranging from process, organization, people, to goods flow related processes, such as sales, purchasing, and manufacturing. Note that many critical project parameters, such as budget and planning, are part of the *internal Operational view*.

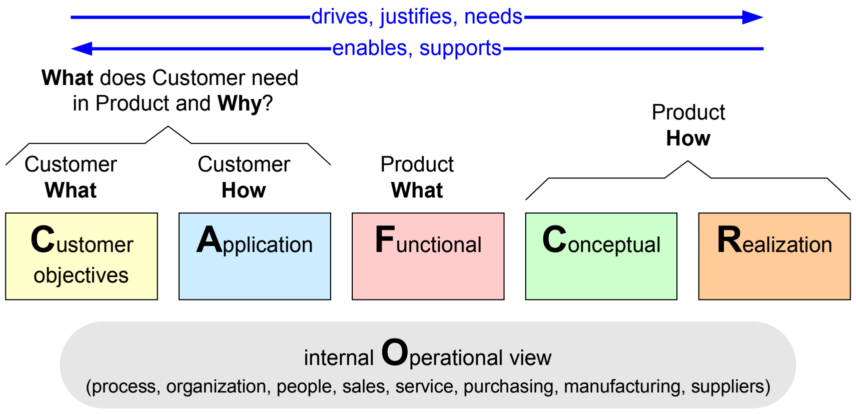
drives, justifies, needs
enables, supports
What does Customer need in Product and Why?
Customer What
Customer How
Product What
Product How
Customer objectives
Application
Functional
Conceptual
Realization
internal Operational view
(process, organization, people, sales, service, purchasing, manufacturing, suppliers)

Figure 2. CAFCR model

The fact-finding process is a quick exploration of all views. The result of this exploration is a broad set of still disconnected data-points: the pieces of the jigsaw puzzle. The set of data-points is far from complete after the first explorations. The main purpose of this bottom-up step is to ground the discussion with facts, to prevent a common pitfall of architecture overviews: a high degree of too abstract and too academic statements.

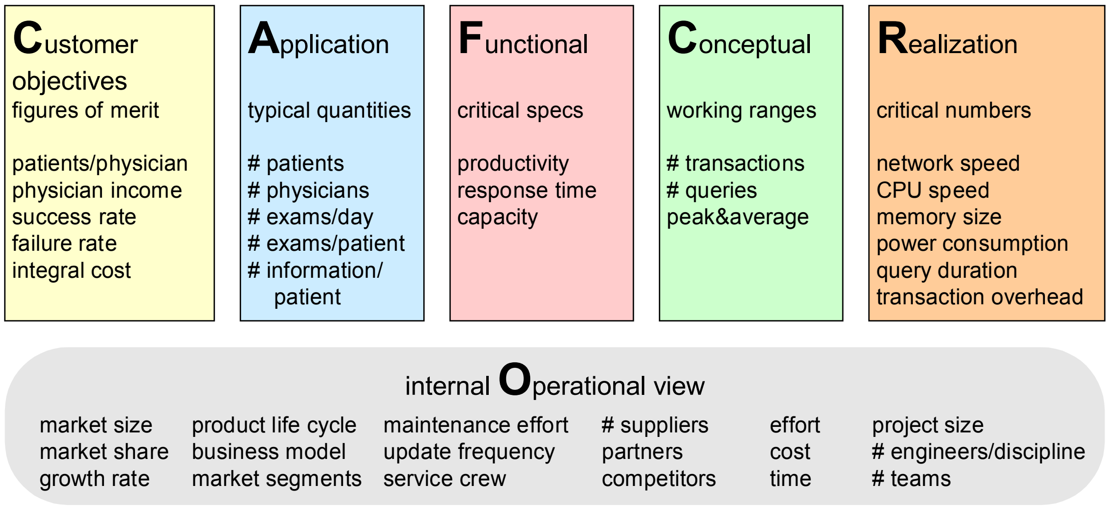

Figure 3. Bottom up fact finding and quantification

In order to achieve this grounding it is strongly recommended to quantify facts as much as possible. Figure 3 shows an imaginary example of an electronic patient record (EPR) service. For all views a number of quantifiable aspects are mentioned that can be addresses during the fact finding.

Extending the overview to the Enterprise level

The project complexity and the related risks increase at least an order of magnitude when project scope increases from single products to departments, to entire enterprises. When scope and complexity increases the overview decreases. This decrease of overview is one of the reasons of project delays and budget overruns. Hence the value of a compact architecture overview becomes even more valuable.

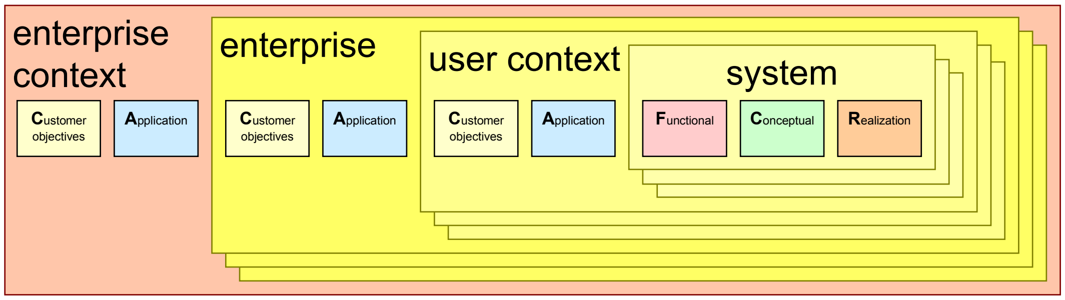

Figure 4. Recursive application of CAFCR model, from system to enterprise context.

For these large scope projects the challenge of balancing breadth and depth is also bigger. The CAFCR model, introduced in Figure 2, uses the simplified notion of *system* and *customer*. However, when the scope increases to enterprise level, then we have to apply the same thinking recursively, as shown in Figure 4. Note that we always have to take the next level of context into account; for an enterprise architecture overview, the enterprise context is part of the overview

Architecture Description and IEEE 1471

Many frameworks and standards exist to create architecture descriptions, see for instance the white paper at the System Architecting Forum (`www.architectingforum.org`) (Muller 2006). In this paper we focus only on IEEE 1471 and CAFCR, although all other frameworks and standards can provide inspiration for creating architecture overviews.

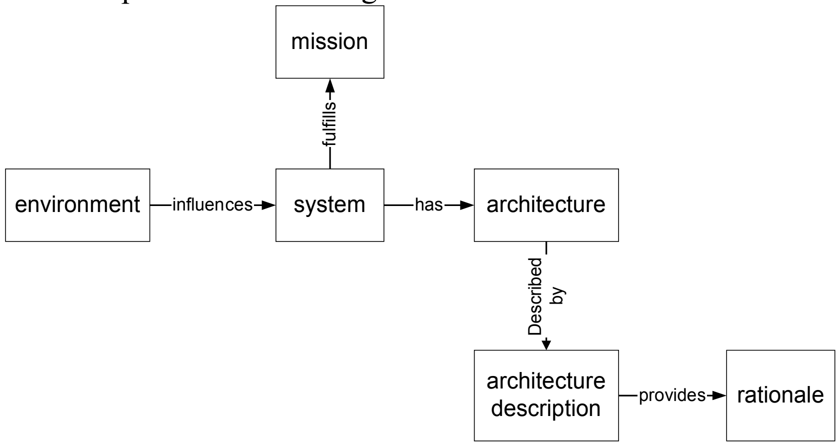

Figure 5. IEEE 1471 top level of the information model

IEEE1471 (AWG 2000) elaborates the notion of architecture descriptions. The top level of the IEEE1471 model positions the system in its environment (see Figure 5). The *environment* is an important aspect for the architecture influencing the system. It also shows that the system has a purpose, called *mission* by IEEE 1471. The architecture description has to provide a *rationale* for the architecture, based on *environment* and *mission* at the one hand and *solution* options at the other hand.

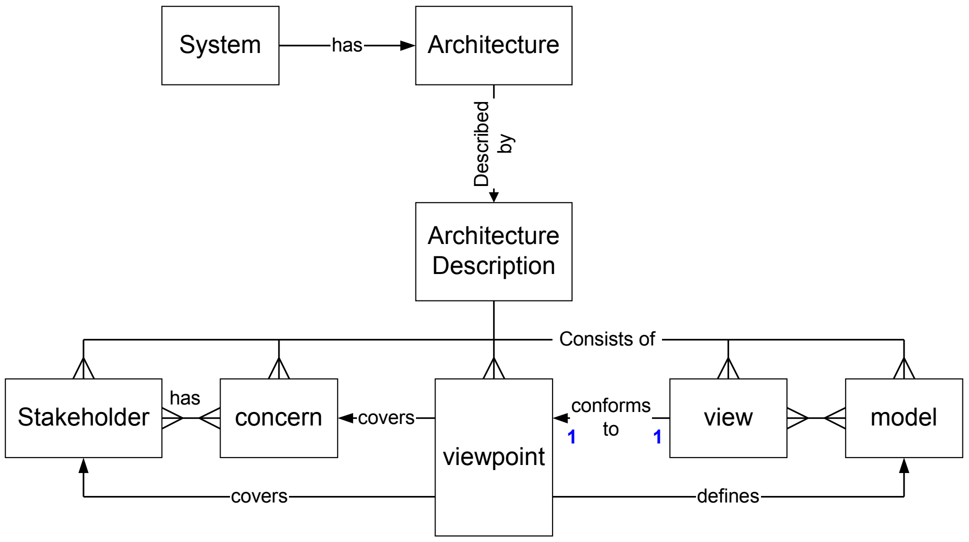

Figure 6: IEEE 1471 simplified information model

The IEEE 1471 framework introduces a number of important concepts to structure the architecture description, as shown in Figure 6:

Stakeholders People or organizations that have an interest in the system under consideration.

Concerns The articulation of the needs and worries of the stakeholders.

Viewpoints The points of view used to describe part of the problem or solution. IEEE 1471 makes a subtle difference between *view* and *viewpoint*. We ignore this difference here.

Models Frequently used method to make problem and solution descriptions.

Architecture description The combination of stakeholders, concerns, viewpoints and models to describe the architecture of a system.

The main contribution of IEEE 1471 is to provide a framework that covers all of these aspects. The individual concepts have been in use by many architects for a long time.

On top of providing the framework, IEEE 1471 also recognizes the fact that complete consistency in the entire architectural description is an illusion. The real world of designing complex systems is full of stakeholders with fuzzy needs, often contradictory in itself and conflicting with needs of other stakeholders. The insights of individual designers are also full of different and changing solutions.

This notion of incomplete consistency is not an excuse for sloppy design; quite the opposite: recognizing the existence of inconsistencies is a much better starting point for dealing with them. In the end, no important inconsistencies may be left in the architecture description.

IEEE 1471 makes another interesting step: it discusses the architecture *description* not the *architecture* itself. The *architecture* is used here for the way the system is experienced and perceived by the stakeholders[1].

This separation of *architecture* and *architecture description* provides an interesting insight.

[1] Long philosophical discussions can be held about the definition of **the** architecture. Many definitions and discussions about the definition can be found, for instance in (Hitchins 1992), (Bredemeyer 2002), or (SEI 2002). Note that we use *architecture description* in this paper, where others use the word *architecture* itself to denote the abstracted and captured artifact.

The *architecture* is infinite, rich and intangible, denoted by a cloud in figure 7. The *architecture description*, on the other hand, is the projection, and the extraction of this rich *architecture* into a flattened, poor, but tangible description. Such a description is highly useful to communicate, discuss, decide, verify, et cetera. We should, however, always keep in mind that the description is only a poor approximation of the *architecture* itself. The overview of the architecture is again only a fraction of the architecture description.

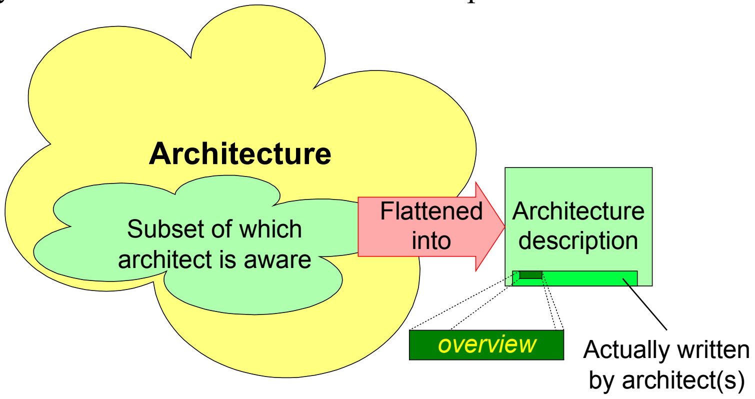
Architecture
Subset of which architect is aware
Flattened into
Architecture description
overview
Actually written by architect(s)

Figure 7: The architecture description is by definition a flattened and poor representation of an actual architecture.

The role of a system architecture description can be formulated as:

- Guiding and constraining framework
- Spanning from opportunity exploration via development, manufacturing to support and retirement
- Supporting communication and decision-making
- Providing an audit trail from problem/opportunity to solution

Presentation and Writing Aspects

The target audience of an architecture overview is a broad set of stakeholders, with different interests, knowledge and backgrounds, see Figure 7. Despite these differences the overview must be *clear* and *easily accessible* for the entire target audience. Note that also the information processing characteristics of the stakeholders differs, some are auditive, some are tactile, and some are visual.

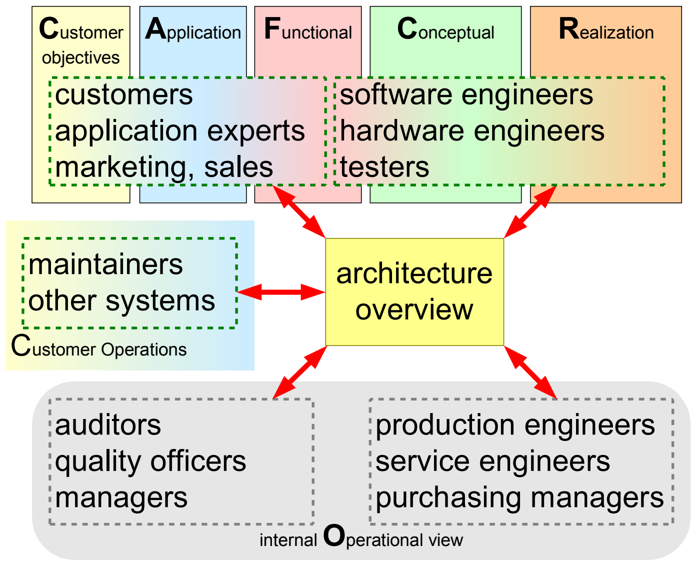

Figure 8. Stakeholders of an Architecture Overview

The architecture overview does not have to be a conventional document. Figure 9 shows a number of forms and mediums that can be used for the architecture overview. The big question mark in this figure indicates that even more creative forms may exist that serve even better in communicating the architecture essence.

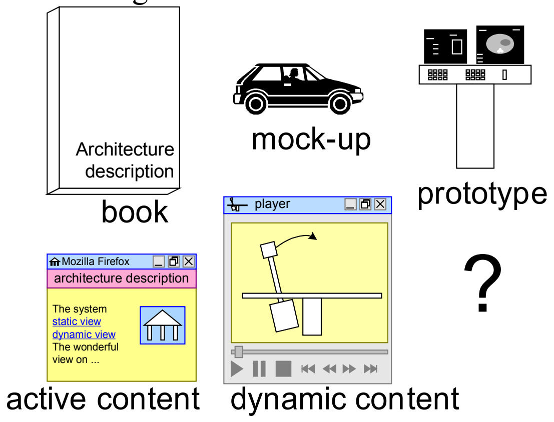

Figure 9. Many alternatives exist for form and medium of the architecture overview

We recommend to provide both visual as well as textual information. Figure 10 shows that the core of the architecture information should be partially visual, as diagrams, tables and lists, and partially textual. The text should explain the visual information and glue the information into a coherent set of information.

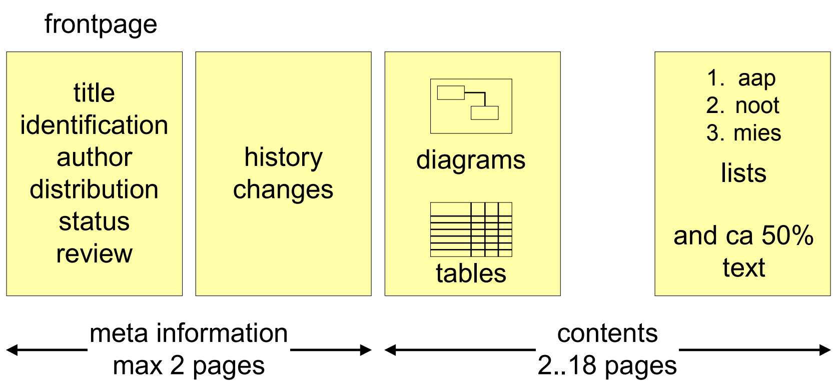

Figure 10. Payload: the ratio between content and overhead

Figure 10 also provides a size indication for an overview. The meta-information of the overview should be 2 pages or less, because the meta-information distracts from the actual architecture. Some meta-information is necessary to manage and understand the overview in the organizational context. The ”payload” of the overview, the real overview of the architecture, should be 18 pages or less. This recommendation is comparable to the reviewable content for a Fagan inspection meeting, see (Eickelmann 2002). If more than 18 pages are needed then this is an indication that the essence of the architecture is not yet articulated. If the overview is too long, then readers have the tendency to skip parts, or worse the total. A long overview is a threshold for reading. Note that the architecture description will be more than the overview only. Relevant architectural information can be captured in other parts of the architecture description.

* Keep your sentences short
* Prefer active verbs
* Use 'you' and 'we'
* Choose words appropriate for the reader
* Don't be afraid to give instructions
* Avoid nominalisations
* Use positive language
* Use lists where appropriate

from Plain English Campaign
http://www.plainenglish.co.uk/plainenglishguide.html

Figure 11. Recommendations from the guide ”How to write in plain English”

The language used in the text should be clear and ”plain English”, see http://www.plainenglish.co.uk/plainenglishguide.html. This guide addresses the aspects mentioned in Figure 11.

There are many possible dimensions that can be used to structure the overview, see Figure 12. Unfortunately, no single dimension is ideal to structure. The structure of the architecture overview must serve its communication purpose. In other words the structure itself is less important than clarity and understandability of the content.

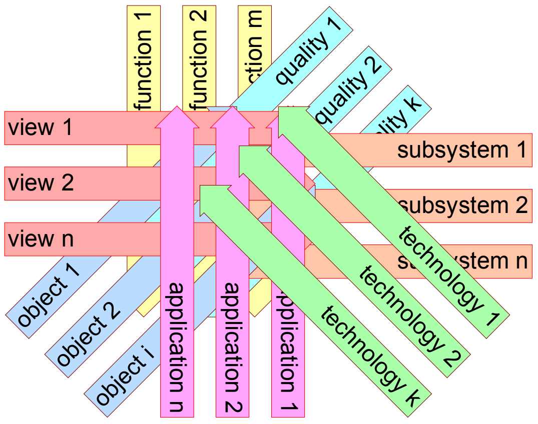

Figure 12: Many possible dimensions to structure the architecture overview

We recommend getting feedback on visualization and text from different potential readers. Not only listen to their immediate feedback, but also observe the impact on them:

- Did the reader understand the essence?
- Did text or visualizations suggest unexpected interpretations?

Lessons Learned

The participants of the Dutch Championship ICT architecture have to provide an architecture description that can be assessed by the jury in less than 2 hours. Contributions to this championship are publicly available at http://www.nkictarchitecture.nl/, although mostly written in Dutch. The members of the jury and several participants experienced the power of these compact architecture descriptions. One of the jury members remarked: "Amazing how quickly you get insight in the project sanity, its risks and the coverage of architecture aspects". In general most contributions for this championship fulfill the overview needs described in this article. In 20 pages or less they provide both breadth (customer environment, business, system context, system specification) as well as depth (relevant system design decisions and their rationale). The nominees of 2004 published an example in the Systems Research Forum (Driessen et al, 2006)

Acknowledgements

This article was triggered by the organization of the Dutch Championship ICT Architecture http://www.nkictarchitecture.nl/. The initiators, Jan Turk, Hans Bot, and Gert Florijn, of this event stimulate the development of the architecting discipline and as such triggered the creation of this article, and provided valuable input and feedback. Erik Philippus sent me the link for the "plain English" guide, the Plain English Campaign http://www.plainenglish.co.uk/ has given permission to use their list of recommendations. Hugo van Leeuwen persuaded FEI to contribute to the NK ICT championship as well as to publish the result in the Systems Research Forum.

References

AWG, Architecture Working Group. *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. The Institute of Electrical and Electronics Engineers, Inc., 2000.

Dana Bredemeyer. *Definitions of software architecture*. http://www.bredemeyer.com/definiti.htm, 2002. large collection of definitions of software architecture.

Nancy S. Eickelmann, Ph.D., Francesca Ruffolo, Jongmoon Baik, Ph.D., Animesh Anant. *An Empirical Study of Modifying the Fagan Inspection Process and the Resulting Main Effects and Interaction Effects Among Defects Found, Effort Required, Rate of Preparation and Inspection, Number of Team Members and Product 1st Pass Quality*, Proceedings of the 27th Annual NASA Goddard/IEEE Software Engineering Workshop (SEW-27'02), 2002.

Derek K. Hitchins. *Putting systems to work*. `http://www.hitchins.co.uk/`, 1992. Originally published by John Wiley and Sons, Chichester, UK, in 1992.

Koen Driessen, Pybe Faber, Kees Kooijman, Pepijn Kramer, Hugo van Leeuwen, Pieter Schroeder, *The FEI Small Dual Beam product Family, Systems Research Forum,* Volume 1, 2006.

SEI, Carnegie Mellon Software Engineering Institute. *How do you define software architecture?* http://www.sei.cmu.edu/architecture/definitions.html, 2002. large collection of definitions of software architecture.

Gerrit Muller. *CAFCR: A multi-view method for embedded systems architecting; balancing genericity and specificity*. http://www.gaudisite.nl/ThesisBook.pdf, 2004.

Gerrit Muller and Eirik Hole. *Architectural descriptions and models.* http://www.architectingforum.org/whitepapers/SAF_WhitePaper_2006_1.pdf, 2006.

Author Biography

Gerrit Muller.

Gerrit Muller received his Master's degree in Physics from the University of Amsterdam in 1979. He worked from 1980 until 1997 at Philips Medical Systems as system architect. From 1997 to 1999 he was manager System Engineering at ASML. From 1999 - 2002 he worked at Philips Research. Since 2003 he is working as senior research fellow at ESI (Embedded Systems Institute). In June 2004 he received his doctorate. The main focus of his work at ESI is on System Architecture methods and on education of future System Architects. Special areas of interest are:

ways to cope with the exponential growth of size and complexity of systems. Examples of methods to address the growing complexity are product lines and composable architectures.

the human aspects of systems architecting (which in itself is a crucial factor in coping with the above mentioned growth)

All information (System Architecture articles, course material, curriculum vitae) can be found at:

Gaudí systems architecting http://www.gaudisite.nl/