Researching Reference Architectures
and their relationship with frameworks, methods, techniques, and tools+

Gerrit Muller[1] and Pierre van de Laar[2]

[1] Embedded Systems Institute, The Netherlands, and Buskerud University College, Norway, Gerrit.muller@esi.nl
[2] Embedded Systems Institute, The Netherlands, Pierre.van.de.laar@esi.nl

+ This work has been carried out as part of the Darwin project at Philips Healthcare under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs under the BSIK program.

Abstract

Reference architectures are seen as one of the means to cope with increased organization size, distributed development, increased integration, increased performance and functionality, and ever faster changes in the market. Our research project Darwin is aimed at evolvability of product families, where reference architecture is one of the research subjects. In this paper we start with positioning reference architectures relative to architecture frameworks, architecting methods and techniques, and tools. Then we focus on our attempts to mine information from past architectures by studying produced artefacts as well as by interacting with the people involved. We explain that it is a long way from detailed facts found in the artefacts to conceptual diagrams that capture the domain essence and that could guide future architectural developments. We illustrate this by discussing two of the smaller research projects in some more detail.

Keywords – reference architectures, analysis, case study, knowledge

1 Introduction

Last year we wrote several papers about reference architectures [3, 9, 10]. The main assertion in these papers is that reference architectures are beneficial for systems that are created in large and distributed organizations. The reference architecture is capturing domain know how from the past and the vision of the future to guide architecting of future systems. The papers were well received; however, a number of questions has been asked repeatedly that have to be addressed. In this paper we will clarify reference architectures in relation to architecture frameworks, architecting methods and techniques, and tools. The following questions were posed multiple times:

1. What is the difference between an architecture framework and a reference architecture?
2. What is the difference between system architecture and reference architecture?
3. What does a reference architecture look like?
4. How do you create a reference architecture?

We do not claim to have all the answers to these questions. A lot of research is needed to answer especially questions 3 and 4 and to (in)validate the assertion that reference architectures are beneficial for large distributed organizations developing evolving product families. We have to realize that while hunting for the proposed reference architectures we did not find yet any example that fits our demanding goals. In other words we are working on a hypothetical entity that still has to be realized, before we can (in)validate its asserted value. However, our research of last year provides some answers that are valuable to share.

We research reference architectures in the Darwin project, www.esi.nl/darwin, as described also in [9, 8]. The Darwin project uses the industry-as-laboratory approach, where the development department of Magnetic Resonance Imaging (MRI) systems at Philips Healthcare is the industrial laboratory. The research of the Darwin project itself is performed by 11 full-time researchers from different universities and disciplines and 3 research fellows from the Embedded Systems Institute (ESI). The main research objective of the Darwin project is to study the evolvability of systems, where the project assertion is that reference architectures facilitate evolvability.

We will first discuss reference architectures in relation with other architectural concepts, such as architecture frameworks, architecting methods, system and product line architectures, to address the first two questions. Next we describe our efforts to create parts of a MRI reference architecture to address the third and fourth question.

2 Reference architectures and other architectural concepts

2.1 Architecture frameworks and architecting methods

In the past, many architecture frameworks and architecting methods have been proposed and are actually used in practice. A well known example of an architecture framework is the Zachman framework [15]. This framework decomposes the architecture description in 6 by 6 views. For all 36 views guidance is provided what and how to present information for that view. A well known architecting method is Structures Analysis and System Design (SASD) by Yourdon [14]. SASD provides a stepwise process and a prescribed set of artefacts to create a system design.

Architecting methods and architecture frameworks have in common that they capture generic information how to create and capture system designs. In both cases a generic recipe is provided with little or no domain information at

all. Both examples originate from the software world. An example of domain information that is still present in both is the emphasis on information models, something that is not relevant for purely mechanical systems for example. However, neither SASD nor Zachman have any knowledge about the addressed applications or services, such as insurance, banking, or air traffic control. Architecting methods and architecture frameworks provide generic guidance for designing a broad set of systems.

The difference between architecting methods and architecture frameworks is the amount of guidance provided for the approach itself. Architecting methods provide an ordered set of steps, the method itself, how to get from the start to a finished design. Architecture frameworks only provide guidance for what information, what presentation and what structure to use to capture the design. Most frameworks, on purpose, are method agnostic. For example, DoDAF [4] is designed to be method agnostic in response to the ongoing feedback that previous military architecting standards were too "heavy".

One of the main deliverables of the architecting effort is an architecture description. Both architecting methods and architecture frameworks tend to focus on the creation of the architecture description. The IEEE standard 1471-2000 [7] is an example of an architecture description standard that is method agnostic. This standard provides a very generic information model to guide the creation of architecture descriptions. Core idea behind this standard is that the architecture description consists of models. More background on the state of architecting can be found in [12]. This white paper describes the current state architectural descriptions and models as discussed in the architecting forum.

The Boderc project [6] used FTMT (formalisms, techniques, methods, models and tools) as framework for the research of Systems Engineering. A cohesive set of FTMT is called a methodology. The previous example SASD is in this definition a methodology, since it also prescribes formalisms and uses tools for support. In fact all these elements that form together a methodology are complementary. *Techniques* use *formalisms* to achieve a desired result; *Methods* provide guidance how to use *formalisms*, *techniques* and *models*. *Tools* support the use of *techniques* and the deployment of *methods*. When we apply methods in practice we also need presentation or visualization guidance in addition to FTMT to support communication and discussion between stakeholders.

In conclusion, we have explained that architecture frameworks and architecting methods are generic, hence not domain specific, means to support the creation and capturing of architectures.

2.2 System and Product line architectures

We will use the IEEE standard 1471-2000 definition for system architecture:

> *The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.*

Note that in this definition of architecture both internal elements of the system (components) as well as external factors (relationship with the environment) together form the architecture. In other words architecture combines understanding of the context with guidance for the design.

System architecture is limited to the system of interest, in relation to its environment, and its evolution. However, when we look at organizations that create systems, then we see that these organizations tend to make many variants of systems, so called product lines or product families, or even complete product populations [13]. These product lines evolve over time and generations of product lines succeed each other. The principles governing the design and evolution of product lines are captured in product line architectures. The more limited system architecture of a single product is derived from and complies with the product line architecture.

We have discussed the different scopes of system architectures and product line architectures or even product population architectures. Essential for all these architectures is that they capture the essential rules governing design and evolution. All these architectures are inherently full of domain information, both from technological nature and from the context.

2.3 Reference architectures

Large organizations that have been creating similar systems and product lines for a long time have a huge amount of knowledge about the domain, both technical, as well as contextual. This knowledge is partially implicit, for example in the heads of designers and engineers, and partially explicit in design repositories and documentation.

In [11], we described that reference architectures get attractive when the organizations become large and distributed. In organizations of few hundred people at one location the social process of information sharing between humans can still be highly effective. However, we have to introduce other, often more formal, ways to capture knowledge and to communicate, when the information sharing process between humans gets less effective. For example, when the organization exceeds the size where this social process is effective or when the organization is distributed over multiple locations, then the communication might become too little to share effectively. Scaling up an organization appears to be very difficult in practice. When we distribute the organization over multiple locations, then sharing of knowledge and communication gets even more difficult. Of course, many systems engineering processes target these problems, for example by defining work breakdowns that allow for cohesive working groups and minimal coupling between the groups. Despite all these measures we observe[1] that larger and more distributed organizations struggle more with sharing knowledge and communication.

We look at the MRI division at Philips Healthcare as example, where the Darwin research is being performed.

[1] Based on personal observation in health care, defense, sub sea, maritime, semiconductor, automotive, consumer, and the information technology domains.

Philips Healthcare has been growing tremendously, partially organically and partially through acquisitions and mergers. In particular, the MRI division has grown from a few persons in the early eighties, to more than thousand persons in 2008. The MRI organization is distributed over many locations, such as Ohio and Florida in the USA, Finland, The Netherlands, Israel, and India. If we estimate the accumulated knowledge of the MRI division since its origin in the early eighties, then we see that several tens of thousands of person years of work has been performed on these systems. Even if part of that knowledge is obsolete, then this organization still has the equivalent of more than ten thousand person years of knowledge accumulation.

We propose reference architectures as a means to capture the essential architectural knowledge over multiple product lines, and multiple generations of systems. The purpose of capturing this knowledge is to provide guidance to architect new product lines and generations. We have observed that in large organizations a tremendous amount of knowledge is available, both implicit in the heads of employees, as well as explicit in design repositories and documentation. However, this knowledge is not structured and captured in ways that facilitate the guidance of next generations. As described in [10] one of the challenges is to reduce the information to digestible proportions, without losing too many essential details.

If we now compare this concept of reference architectures with architecture frameworks, then we assert that reference architectures are inherently rich in domain information, technical and contextual. In that sense reference architectures are quite similar to system architectures and product line architectures. The main difference between reference architectures and system or product line architectures is that reference architectures have to abstract even more from implementations; reference architectures are not system or product line specific.

Over time an organization has built up experience about many domain specific problems and many possible solutions in many different circumstances. In the software world the combined knowledge of problem, solution and circumstances is captured as pattern [5], based on Christopher Alexander's patterns [1]. The idea behind reference architectures is to capture this type of knowledge, for example as patterns. It is important to realize that most of this knowledge is domain specific.

So, we are not so much looking for generic patterns, such as the observer pattern. Instead we are looking for domain specific patterns, such as, for example, guiding principles for the RF (Radio Frequency) transmission. The excitation of nuclei as part of the MR imaging is done by RF pulses. To shape and actually generate the RF-field many technical solutions are available, with their advantages and disadvantages. The most relevant guiding principles for the RF transmit chain can be captured in the reference architecture. Guiding principles to shape and generate RF fields are highly domain-specific. These principles have evolved slowly in the last few decades. The slow evolution makes it attractive to capture the principles explicitly, because then they can be re-used with little maintenance effort.

3 Research in the Darwin project

The full-time researchers of the Darwin project have in general an academic background. These researchers are not seasoned MRI designers, but independent scientists. The consequence is that they start with an empty sheet of paper. The benefit is that the researchers are unbiased, while the obvious disadvantage is that a significant learning investment has to be made to be able to contribute. Roughly the following approaches are being followed by the researchers to search domain knowledge that could be incorporated in the MRI reference architecture:

1. Analysis of the repositories and the meta-information.
2. Observation and analysis of running systems.
3. Reading of documentation.
4. Interviewing of stakeholders.
5. Workshops with researchers and stakeholders.

Every researcher tackles a specific case to make the research scope manageable. For example, some of the researchers look at the evolution of clinical application packages. For that purpose software dependencies between applications and other building blocks of the software are identified. An individual researcher typically has to limit the scope and has to connect to the researcher's prior discipline. The organizational project perspective is that researchers define smaller projects in time and in human resources where they cooperate with a limited number of Philips designers or architects. The idea is that the combined set of smaller projects over time advances the insight in evolvability and that parts of the MRI reference architecture will be created.

Most researchers have been working on the project now for about 2.5 years. In the beginning most researchers have explored the current MRI architecture from their disciplinary perspective. Since we are now halfway the research project we have challenged all researchers to create one view that according to their current insights would belong to the MRI reference architecture.

3.1 Analysis tools

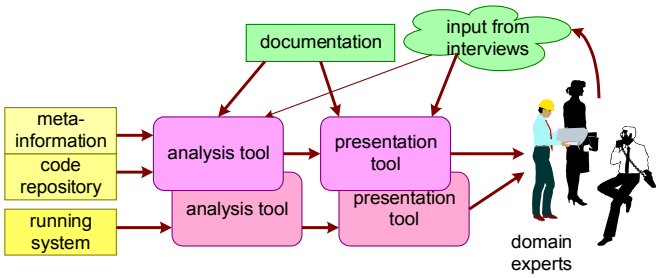

Figure 1 - *Typical research activity: analyzing code repository or running system and presenting relevant information to Philips stakeholders.*

Figure 1 shows a typical research activity, where analysis tools operate on repository data or observations from running systems. Analysis tools should support the mining of knowledge from existing realizations. Repository data provides static information about the realizations, while running systems provide a snapshot of dynamic information of a limited set of events on one instance of the realizations. The presentation of the results is based on domain understanding derived from documentation and interviews. The output of the presentation tool is not yet a reference architecture view. For example, the output might show mutual dependencies between subsystems or components, or the typical data and event flow in a running system. This output might trigger domain experts, for example when subsystems are much more dependent than expected by the experts, or when the data flow is more complicated than expected.

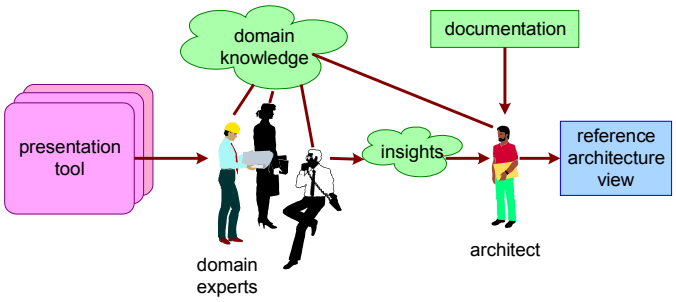

Figure 2 - *Quite some steps are needed to transform output of analysis tools into a reference architecture view.*

The output of analysis tools and the related interaction between domain experts with their implicit domain knowledge can trigger new insights. An architect can transform these insights, using implicit domain know how and explicit documentation, into a view of the reference architecture. The interaction process and the architect's involvement should not be random, but need based. For example, the introduction of new application features or new technologies might drive the interest in reference architecture views. These views will then be related to the entities well known to the MRI architects, such as clinical packages (a package of software providing a single clinical application), building block dependencies (the local variant of components) and the execution architecture. An example of such view is provided in Figure 3. The researcher who produced this view started his research by building and using a set of analysis tools. The results from (potential) users perspective was quite disappointing. The researcher chose a different approach: he joined an engineering team working on clinical packages. The researcher applied his analysis tools in this approach when needed. His contribution is now highly appreciated by the organization. Based on these experiences the researcher produced Figure 3 as draft view for the MRI reference architecture.

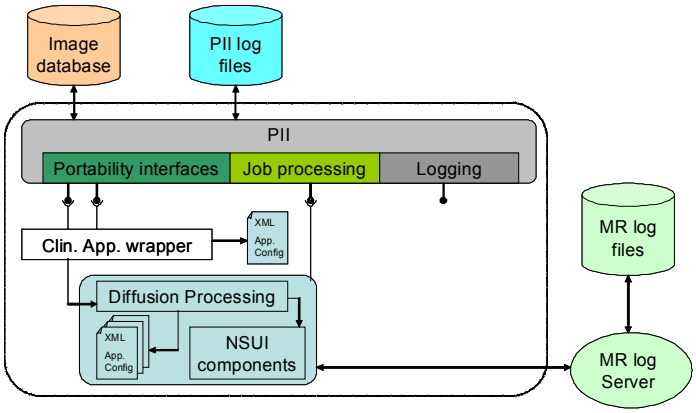

Figure 3 - *Draft reference architecture view for clinical packages, based on analysis output and interaction with domain experts.*

Most researchers have been working on the activities depicted in Figure 1. Our challenge to translate their own insights into a reference architecture view forces them to move more to the transformation process shown in Figure 2. The original idea behind analysis tools is that the design repositories, such as the software code repository with millions lines of code, contain lots of design knowledge. Unfortunately, this knowledge is the result of a process where few user needs via specification and design steps finally are transformed in detailed realization steps in software code. The analysis tools are part of a reverse architecting effort. However, Figure 2 shows that the analysis tools have to be combined with other knowledge to be able to turn detailed repository information into reference architecture views.

As example Figure 4 shows the presentation of run-time analysis tools. [2] gives a more elaborated description of this research. Based on discussions with experts it was decided to analyze the structure of all RF-coils (RF = Radio Frequency) used to receive the MR signal. The system architects did have an interest in the system design of the RF-coils and all related interfaces, both hardware and software. The main entities in the system design are building blocks, tasks, and processes. Building blocks are the units of decomposition in the software repository; The building block decomposition is static. Processes are the units of the decomposition for concurrency; The process decomposition is dynamic. Tasks are the decomposition of the work to be done by the system. At the most detailed level of engineering these entities are realized in code and data.

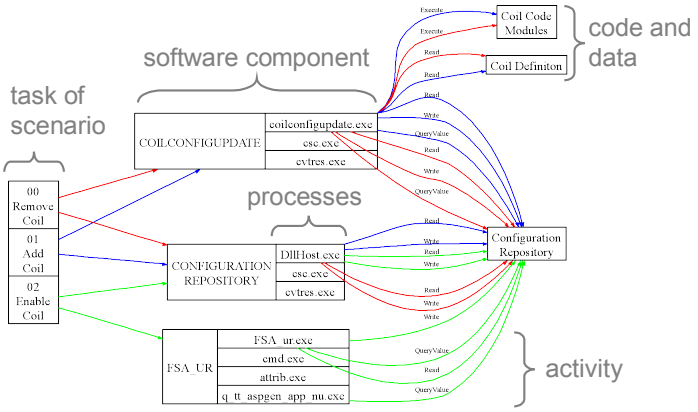

Figure 4 - *Presentation of run-time analysis of RF-coil related tasks.*

The run-time analysis tool analyzes for one scenario what building blocks, processes and tasks are involved and how this maps on code and data. A scenario is some user level operation, such as adding a new RF coil to the system. This tool lifts the analysis from lines of code and individual data elements to the conceptual level of entities being used by designers. It helps the designers to relate the dominant static view of the system to the actual dynamic behavior. The limitation of this approach is that it operates on a set of scenarios (out of thousands of relevant scenarios) on one realization in one system configuration. In other words the tool zooms in on one specific use, and creates insight at higher conceptual level for that specific use. By sampling different scenarios and configurations the broader conceptual picture is created.
We have discussed two examples of research with analysis tools. One example used static analysis tools (approach 1) to study clinical packages. The other example used run-time analysis tools (approach 2) to study RF coil interfaces.

3.2 Reflection on analysis tools in relation to reference architectures.

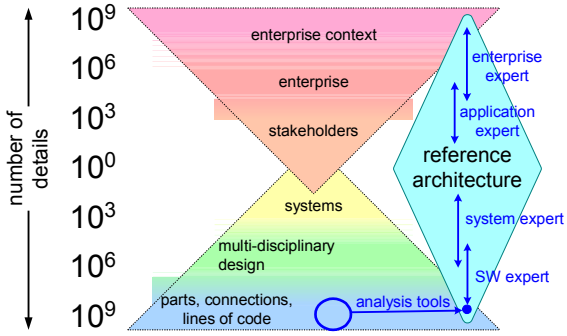

Figure 5 - *Positioning research of analysis tools in the "abstraction diabolo".*

In [10] we discussed the level of abstraction and the number of details of reference architectures by a diabolo-like figure. Figure 5 is the same figure overlaid with the reference architecture itself, the analysis tools and a number of stakeholders. The diabolo part of the figure shows that typical collections of systems are defined by billions of details of mono-disciplinary nature: mechanical characteristics of parts, connections and components, software lines of code et cetera. During the multi-disciplinary design we use less detailed abstractions. At system level we specify systems even more compact with thousands of system characteristics. These systems operate with many stakeholders, each with a significant set of characteristics. These stakeholders operate within an enterprise, which is orders of magnitude more complex, with large numbers of employees, and systems; both with large number of characteristics. The enterprise again operates in some broader context, where the number of details again is orders of magnitude larger. Reference architectures have to be limited in size to be useful and manageable, so most information is at higher abstraction layers in the middle of the diabolo. However, the art of creating usable reference architectures is to be able to identify essential details and to capture those in the perspective of the "big picture".
The overlay of Figure 5 shows that analysis tools typically analyze highly detailed repositories. The purpose is to find these essential details. However, the assessment of the relevance of details depends on all higher layers: multi-disciplinary design, system specification, stakeholder needs, enterprise needs and enterprise context. Analysis tools may find candidates for essential details that can be assessed by stakeholders. Good analysis tools result in few false positives (details that turn out not to be essential) and few false negatives (missing details that are essential, but that are not found by the tool).

3.3 Interviewing, reading documentation, and workshops

[9] described an example of functional and physical architecture diagrams made to study the evolution of the communication technology internal in the system. These diagrams are getting close to reference architecture views. In one of the workshops of researchers and designers from Philips (approach 5) the quantification of these figures was discussed. Quantification in terms of figures of merit, for example, would bring these diagrams closer to reference architecture views. This activity is planned in the near future.
The researchers did dive more in specific interfaces, related to RF-coils, of this part of the system before elaborating the quantification. These interfaces were of particular interest for ongoing discussions about the system design. Interviewing (approach 4) was chosen as main approach in addition to reading documentation (approach 3) and looking for information via the previously discussed analysis tools (approaches 1 and 2). The interview approach turned out to be difficult, due to organization size and all multi-factors [12] that trigger the research of reference architectures.

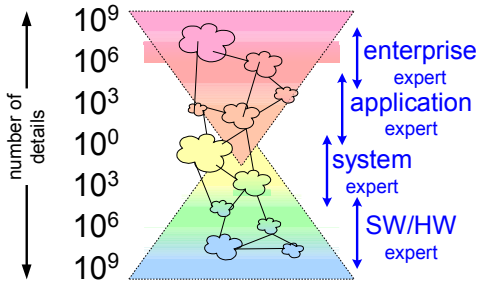

Figure 6 - *Positioning interviewing results in the "diabolo".*

The interview results are scattered across the entire diabolo, as visualized in Figure 6. What makes it more difficult is that the results are not always cohesive and sometimes simply contradictory. The explanation of this outcome is that the interviewed stakeholders have different concerns and perspectives. This is the normal architecting challenge: to find an acceptable and appropriate solution in a non-trivial socio-economic force field. The initial outcome of such a set of interviews is a map with technical facts, business propositions, operational considerations, opinions, concerns et cetera.
The interview results have been transformed into a set of more structured diagrams, ranging from the process flow with stakeholders, their needs and their responsibilities to data flow diagrams in the software control part of the system. To transform these diagrams into the intended reference architecture level we have to make several steps. The current diagrams describe a status quo. These diagrams might be far from complete; we might have to study more configurations and situations. The next step is to reduce the diagrams to the essentials, an abstraction step. The last step is to include future needs and vision to provide guidance for the future.

3.4 Current Research Status and Future Research

We have discussed the search for domain knowledge in four complementary ways: (1) static analysis, (2) run-time analysis, (3) reading documentation and (4) interviewing. We have had several workshops (approach 5), but they did not yet have the direct focus to work towards reference architecture views. The research is performed mostly by relative outsiders, unbiased by years of history. None of these four ways so far has produced reference architecture views. Intense interaction with domain experts and architects is required to transform the current research results, such as the output of analysis tools and many diagrams, into reference architecture views. Two examples that we discussed focused on interfacing RF-coils, providing useful and complementary information. However, together these activities do not yet provide the RF interface part of the MRI reference architecture, because the information is still too fragmented and detailed. The produced information has to be integrated further and reduced to the essentials to get to the level we intend the reference architecture to be.

We will have to intensify the interaction of researchers and stakeholders to get closer to reference architecture views. Approach 5, workshops with researchers and domain experts, deserves more research effort, because this method might help to lift the abstraction level of the information gathered by the other approaches. The senior research fellows will make an attempt to create some reference architecture views based on the available research results, by working on actual cases with domain architects. For example by combining all RF coil interfacing work and intense interaction with RF coil experts and stakeholders.

4 Summary and Conclusion

We have discussed the potential value of capturing domain specific architectural knowledge, especially for large and distributed organizations. Reference architectures are capturing this architectural knowledge to provide guidance for future architecting efforts. The main difference between architecture frameworks, architecting methods and reference architectures is the degree of domain specificity; reference architectures are highly domain specific, frameworks and methods capture the generics.
Our research with about 15 researchers tries to penetrate this pile of accumulated knowledge with tens of thousands of person-years effort and with more than thousand people working on it continuously. We have chosen a set of five complementary approaches (analysis, reading, observation, interviewing and workshops). We make progress, but at the same time we conclude that we still have to bridge a significant gap between analysis tools and interview-based methods to reference architecture views. More research is needed to answer the questions:

- What does a reference architecture look like?
- How do you create a reference architecture?

Answers of these questions are a prerequisite to (in)validate the asserted value of reference architectures.

5 Acknowledgments

Pierre America, Teade Punter and Dave Watts provided feedback. All researchers from the Darwin project contributed with their research work, especially Trosky Callo with the run-time analysis and Pieter van der Spek with clinical packages view.

6 References

[1] A Pattern Language: Towns, Buildings, Construction. By Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, Shlomo Angel (1977) New York: Oxford University Press. ISBN 978-0195019193.

[2] Analyzing the Actual Execution of a Large Software-Intensive System for Determining Dependencies, by Trosky B. Callo Arias, Paris Avgeriou, and Pierre America at WCRE08 in Antwerp, October 2008.

[3] The Concept of Reference Architectures, by Robert Cloutier, Gerrit Muller, Dinesh Verma, Roshanak Nilchiani, Eirik Hole and Mary Bone, submitted to INCOSE Journal of Systems Engineering, 2008.

[4] DoD Architecture Framework, Volume 1: Definitions and Guidelines, Version 1 US Dept. of Defence, 2003.

[5] Design Patterns: Elements of Reusable Object-Oriented Software, by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides (1995). Addison-Wesley. ISBN 0-201-63361-2.

[6] Boderc: Model-based design of high-tech systems, edited by Maurice Heemels and Gerrit Muller, Boderc Symposium 2006, published by Embedded Systems Institute, Eindhoven, http://www.esi.nl/publications/bodercBook.pdf.

[7] IEEE Recommended practice for architectural description of software-intensive systems, by the Institute of Electrical and Electronics Engineers, Inc., IEEE Std 1471, 2000.

[8] The Darwin Project: Evolvability of Software-Intensive System, by Pierre van de Laar et al, ICSM 2007, Paris, October 2007.

[9] How Reference Architectures support the evolution of Product Families, by Gerrit Muller, CSER 2008 in Los Angeles

[10] Right Sizing Reference Architectures; How to provide specific guidance with limited information, by Gerrit Muller, INCOSE 2008 in Utrecht.

[11] Reference Architectures; Why, What and How, edited by Gerrit Muller and Eirik Hole, white paper from the System Architecting Forum, March 2007,

[12] Architectural Descriptions and Models, edited by Gerrit Muller and Eirik Hole, white paper from the System Architecting Forum, March 2006,

[13] Building product populations with software components, by Rob van Ommering, ICSE 2002 in Orlando, Florida.

[14] Modern Strutures Analysis, by Edward Yourdon, Prentice Hall 1989.

[15] The Zachman framework for enterprise architecture, by John Zachman, http://www.zifa.com/, 1987.