This is an English translation of a German conference paper presented at the 2012 conference of the German "Gesellschaft für Systems Engineering e.V.". Please cite this document as follows: " M. Dänzer, W. Gerritsen, J. G. Lamm and T. Weilkiens, "Funktionale Architektur trifft Schichtenarchitektur", Tag des Systems Engineering, Paderborn, Germany, November 2012.. English translation by J. Lamm."

Functional Architecture Meets Layered Architecture

Matthias Dänzer[1], Werner Gerritsen[1,3], Jesko G. Lamm[1,3], Tim Weilkiens[2,3]

[1]Bernafon AG, Morgenstrasse 131, 3018 Bern, Schweiz,
www.bernafon.com / Corresponding author: jla <at> bernafon.ch

[2]oose Innovative Informatik GmbH
www.oose.de / Corresponding author: tim.weilkiens <at> oose.de

[3]Functional architectures working group of GfSE, German chapter of INCOSE
fas@gfse.de

Abstract: This paper presents a novel approach for modeling layered architectures in SysML and describes how to combine it with the existing FAS method in modeling functional architectures. This is demonstrated based on an example from the hearing instrument domain.

1 Introduction

Layered architectures are commonly used and recognized in software development. According to the authors' experience, the strict separation of layers improves maintainability and testability of software-intense systems. This could be seen as a matter of the software architect only, with no relevance to system architecture. By contrary, we believe that the layered architecture for software systems should be anchored in system architecture, in order to have a common layered architecture of all the (software) components in the system, enabling to easily connect them and to be able to foresee Test Access Mechanisms [LEB09] for the system on system architecture level and optimize them according to the layers of software architecture.

There are cases in which system architects describe systems by functional architectures [LW10, KLW11], which are based on the approach of decomposition and are therefore not directly compatible with layered architectures [MR02].

This paper shows how functional architectures and layered architectures can be modeled and interconnected. Based on the analysis which Maier [Mai06] did regarding the relationship between hardware architecture and layered architecture, we also take views on physical components of the system into account and show how they can comply with our approach (in a way that differs from Maier's ideas).

Architectural modeling, as shown in the following, works for any kind of cross-domain system development. Nevertheless, we limit the scope of layered abstractions to information processing subsystems, like e.g. software components, dedicated logic and communication units.

We have chosen OMG SysML™ [OMG10] as a modeling language, because it enables the modeling of multiple views on the system (e.g., functional views and views on layered architecture) and there is a method (FAS method) for modeling functional architectures according to [LW10] that has been used successfully with SysML [KLW11].

The modeling approach we propose is used today for product development in the hearing instrument domain; for simplicity, however, we demonstrate it based on a simplified example model. This model will be presented after some definitions of terms and an introduction to the modeling approach, in order to then use it for demonstrating the approach by means of the example. During the presentation of this example, the corresponding procedure is exemplified; it is however not generic enough to be called a generally applicable method.

2 Definitions

A *layer* groups system elements of equal abstraction level. An element's abstraction level cannot be quantified in an objective way and is a matter of the architect's judgment. For assigning names to abstraction levels, we use the *layer category*, e.g., "Transportation Layer" or "Application Layer" in the communications systems domain.

A *layer model* is a hierarchical distinction of elements, with the purpose of encapsulation that enables the definition of well-defined interfaces between layers. The elements of a lower layer are not part of a higher layer; the higher layer's element can rather access services of the lower layer via interfaces that have been defined by System Architecture („is-used-by", not „is-a-part-of" [Mai06]). The distinction of layers results from assigning elements to levels of abstraction.

Architecture is the thought that structures a system under development by its elements and their connections.

Functional architecture is architecture using functional elements, which are based on system's functions. More elaborate definitions around functional architectures can be found in [LW10].

Layered architecture is architecture based on a layer model that can be described by that model's interfaces. The layers are its elements.

Physical architecture is architecture based on *physical Element*, which represent parts of implementation that exist in reality.

3 Modeling Approach

3.1 Modeling Functional Architectures with SysML

There is a method (*FAS method*) according to [LW10], deriving a structural architecture of functions from use cases that are refined with activity models. The most outstanding concepts of the FAS method are its focus on use cases – and thus on the user – and the structural view on system functions. The FAS method is described in [LW10]. It will be briefly outlined here, and we describe how this method that is in general independent of the modeling language can be used with SysML.

Use cases of the system are refined by means of activities. Their structure can be represented by trees, which describe call hierarchy as well as the relation to input and output objects [OMG10]. Heuristics (e.g. according to [LW10]) together with the system architect's experience guide the process of assembling functional groups by grouping activities from activity trees based on criteria of cohesion. Based on the functional groups, the corresponding functional blocks will be created. They are the foundation of the functional architecture. In SysML they are modeled by blocks of stereotype «functional block». They can be interconnected and interfaces can be assigned to them if their functions interchange objects. The result will be a SysML model of the functional architecture.

3.2 Modeling Layered Architectures with SysML

We propose to model layer categories as abstract SysML blocks with stereotyope «Layer» and to model the layers themselves as normal blocks of same stereotype, with a generalization relationship towards the abstract block representing the layer category. For example, an abstract block "Transportation Layer" with stereotype «Layer» could model the category of all transportation layers, whereas a derived concrete block „DECT Transportation Layer" (where DECT is a cordless telephony standard) of same stereotype would describe the dedicated transportation layer of the DECT communication protocol (figure 1). A layer can be decomposed into sublayers. We do this using the composition relationship.

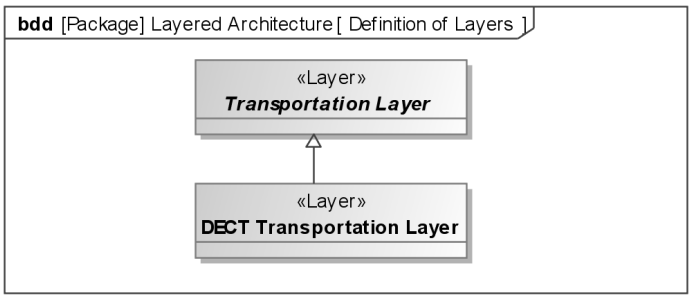

Figure 1: Layer Category "Transportation Layer" and concrete "DECT Transportation Layer" in SysML. An italic block name means that the block is abstract.

3.3 Traceability

SysML enables tracing elements to requirements with «trace» relationships, to express that the modeled aspect of the system is relevant for satisfying the corresponding requirement. Hence, traceability from solutions back to requirements can be established by means of the corresponding connections from functional, physical and «Layer» blocks to requirements.

To keep the amount of work for this very demanding task within reasonable bounds, we need conventions that help limiting the number of resulting «trace» relationships. By addressing this need together with similar topics around the «allocate» relationship that enables traceability between different views (e.g., the functional and physical ones [LW10]), we come to the following proposal of conventions:

- «trace» relationships that point to functional requirements should originate from functional blocks. But before introducing this kind of traces, consider if the traceability to and via use cases that comes with the FAS method is sufficient.

- Non-functional requirements should be treated like functional requirements, if possible (i.e. if referring to a function of the system) or should be traced to the corresponding functional requirement. Otherwise a «satisfy» relationship should originate from a physical block or a «Layer» block (e.g., such a relationship could point from a physical block “housing” to a requirement “housing color”, which certainly does not match one distinct system function).

- Functional blocks can be linked with physical blocks by means of an «allocate» relationship [LW10]. It is recommended to connect the corresponding “part properties” – and not the blocks themselves.

- It is again the «allocate» relationship that can be used to link the part properties that result from functional blocks with the ones resulting from «Layer» blocks, where the direction should be the same as towards the physical block, when looked at from the perspective of the functional block. It is recommended to split functional blocks in such way that the resulting blocks can each be linked to one unique layer. Functional blocks that have been created only for the corresponding reason of having a unique mapping to layers, and not based on domain knowledge, should be marked. To do so, we assign the stereotype «layered» and concatenate the block name with the layer name (figure 2).

- Finally, part properties that result from «Layer» blocks can be linked with physical blocks by means of an «allocate» relationship, where the direction is the same as in the mapping to a functional block, seen from the perspective of the physical block.

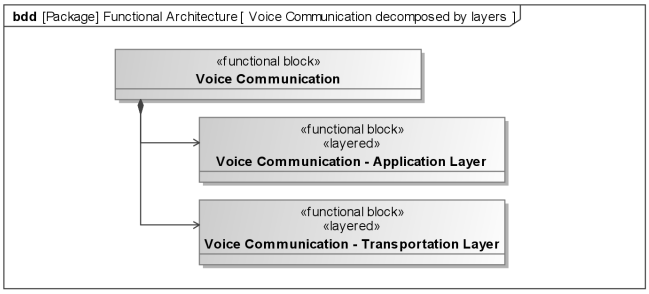

Figure 2: The stereotype «layered» marks layer-specific functional blocks

If not only blocks but also interfaces (connector elements) are admitted as endpoints of «allocate» relationships, then additional conventions are needed. Here is our proposal: If a connector from within the functional architecture is mapped to a part property that results from a «Layer» block, then this shall express that the interface the connector represents is implemented in the corresponding layer and is not visible from the outside, due to the encapsulation that is established with a layer. By contrast, if a connector from within functional architecture is mapped to a connector inside the layered architecture, then this should denote that the corresponding interface in functional architecture is implemented in an interface between layers and is thus visible from the outside.

4 Example System

The example to be used is a simplified system from the hearing instruments domain. One to two hearing instruments per hearing instrument user work primarily as an amplifier. Amplification is applied either to sound input or to a telephone output that is e.g. transmitted wirelessly to the hearing instrument. It is possible to select from the input signals by means of manual switching. Furthermore, the volume can be adjusted. Depending on which variant of the product is used, the corresponding adjustments are triggered via controls that are mounted on the hearing instrument itself, or via a wireless signal from the other hearing instrument after this device has been operated by a control, or by a remote control. The different variants of controlling the device are relevant no earlier than in the step of describing the physical architecture. This is why they we distinguish them neither in use cases nor in the functional architecture. Figure 3 shows the corresponding use cases by means of a SysML use case diagram.

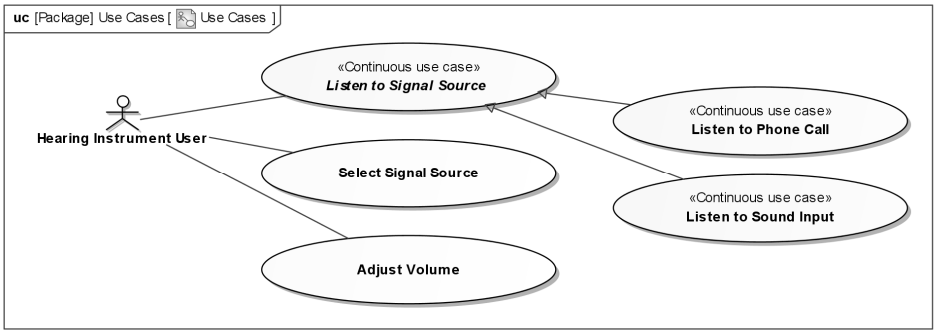

Figure 3: Use cases of the example hearing instrument system

5 Applying the Modeling Approach to the Example

5.1 Procedure

During the work on the subject of this paper, the example system and its architecture were worked on in parallel to finding the right modeling approach, and they were used for validating the approach. The work was done mainly by a team of architects, like it is typically assigned to real projects. As one of the tasks, the team developed the functional architecture of the example model, according to the FAS method. The use cases according to figure 3 and all other model representations in this paper a result from that team work.

In order to follow the ideas from [KLW11] for automatically synthesizing a model that describes the functional architecture, we used the FAS plugin, which is provided by the FAS working group of GfSE for different modeling tools [FAS12a]. Here, the FAS plugin's version for the MagicDraw® tool [FAS12b] was chosen.

5.2 Refining Use Cases by Means of Activity Models

The use cases according to figure 3 were refined by multiple activity models. One activity diagram was created per use case, with two partitions – one of them having the «I/O» stereotype from the FAS plugin. During the modeling of activities, input / output operations were separated from core operations of the system. Activities that contribute to system input / output were assigned to the «I/O» partition. Figure 4 shows the result, based on the use case "Listen to Signal Source". The fact that this figure resembles the corresponding diagrams from [LW10, KLW11] is a consequence of choosing a quite similar example and similar use cases.

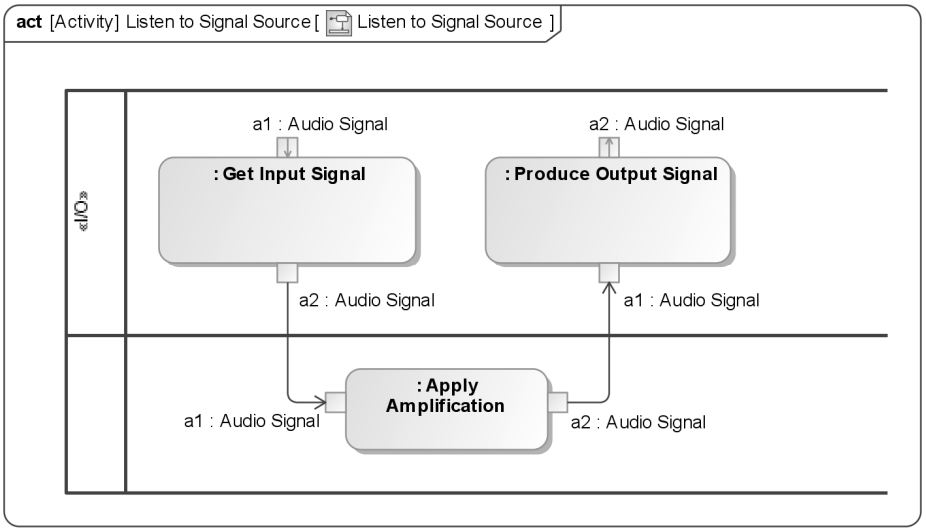

Figure 4: Activity model of use case „Listen to Signal Source"

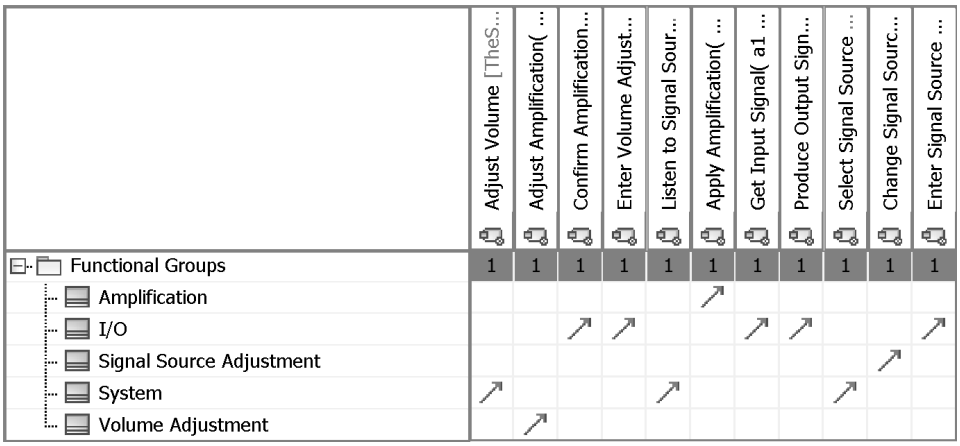

Figure 5: Functional groups for the grouping of activities

5.3 Grouping Activities into Functional Groups

Functional groups were modeled as blocks with stereotype «functional group». In order to assign activities to a group, the «trace» relationship was used in the direction of ascending abstraction level (i.e. from blocks to activities). To create those relationships in the tool, its matrix representation feature was used. First, the script "Initialize functional groups" [FAS12c] of the FAS plugin was called, in order to have it automatically perform the two steps of first creating two functional groups "System" and "I/O" and then linking these to the corresponding activities. Then, manual grouping and regrouping was done until all activities were assigned to functional groups uniquely and with an adequate grouping. Figure 5 shows the resulting assignment.

5.4 Modeling the Layered Architecture

The layered architecture was constructed with three categories, user interface layer, logical layer and technical layer (these are, in our opinion, typical categories for layered architectures). The result is shown in figure 6: on the left hand side, layers and their categories are shown; on the right hand side the figure shows the system context diagram of the layered architecture, indicating that the different layers are connected with each other – but also with the actors – via interfaces (connectors). The connections between the technical layer and the actors are special cases. They model the direct signal flow from the sound environment via transducer and amplification blocks of the hearing instrument to the user. This model representation has been chosen, because it seemed impractical to define a "logical" view or even a "user" view on sound.

Already before completion of the layered architecture, the functional architecture had been automatically generated from the functional groups according to section 5.3, by means of the FAS plugin for MagicDraw®. During the allocation to the layered architecture, it was noticed that the "I/O" block had been scattered over more than one layer, due to the above-mentioned special cases related to sound. Therefore, the model of input / output blocks was post-processed, resulting in a refined granularity: the "I/O"

block was split into different blocks, based on the nature of the different information flows. As a result, the new blocks “I/O Program”, “I/O Volume” and “I/O Sound” were created. In consequence, a unique assignment of the block “I/O Sound” to the technical layer was possible. Here is the detailed procedure that was carried out: first, the functional groups from 5.3 were adapted and the functional architecture was once more created automatically; second the new functional architecture was altered manually for the first time – the “Amplification” block was refined in terms of logical and technical functionality, by creating the corresponding sub-elements “Amplification – Logical” and “Amplification – Technical”. This enabled a unique assignment of functional blocks on the finest level of granularity to layers of the layered architecture. The resulting functional architecture is shown in figure 7. It could be allocated to the layered architecture, as shown in figure 8.

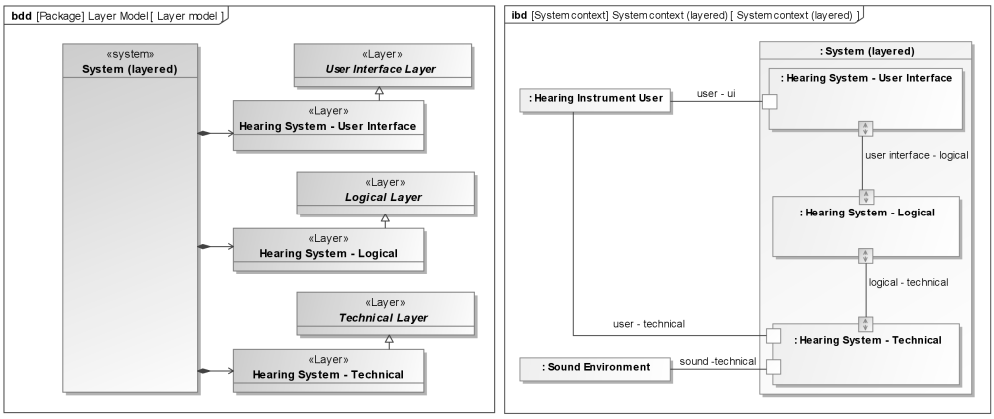

Figure 6: Layer model with layer categories (left) / with interfaces (right)

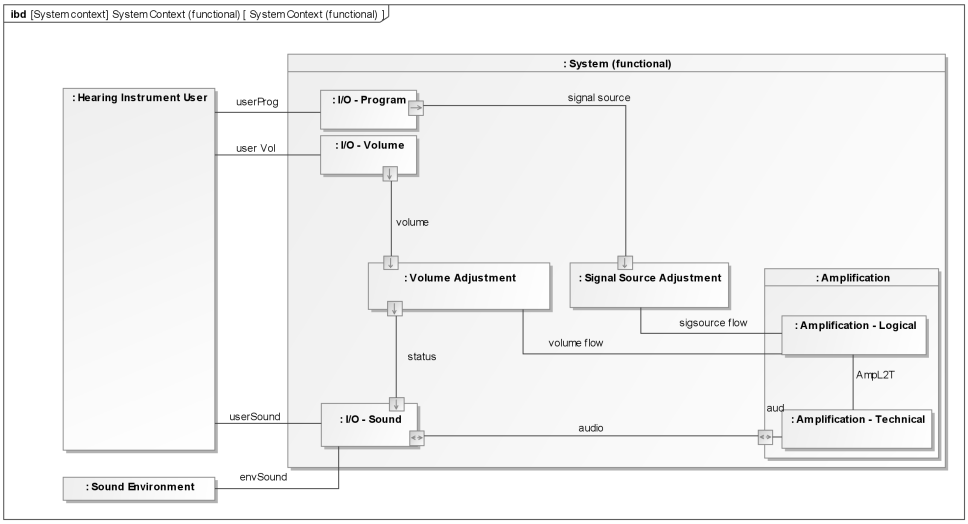

Figure 7: Functional Architecture with a subdivision by layer in „Amplification“

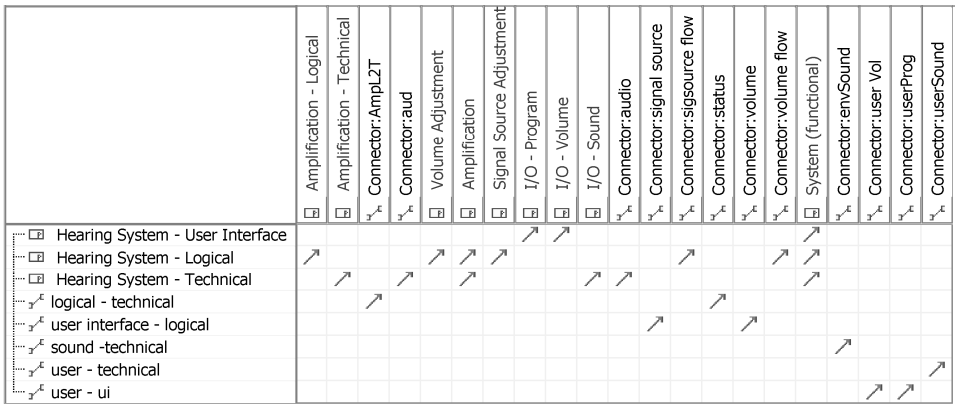

Figure 8: Allocation between the layer model and the model of functional architecture

6 Observations and Discussion

One observation was that the automation obtained by the FAS plugin facilitates iterative procedures, because the plugin allows for quick re-synthesizing of functional architecture and thus enables quick repartitioning of functionality between functional blocks. In the given case, the I/O block had to be subdivided, because parts of the input / output operations were described on a different abstraction level than the rest of them and were thus assigned to a different layer.

It became apparent during the work on this paper – but also during daily business of some of the authors – that the effort for modeling a layer-oriented functional architecture is considerable, because the approach we show here lacks tool support with respect to layered architectures, even though automation for the creation of functional blocks is available. The final model provides an excellent overview of the system at the price of considerable modeling effort.

7 Conclusion and Outlook

The paper has presented an approach for modeling functional architectures and layered architectures together in SysML, and demonstrated the approach by means of an example. It has been observed that the approach provides an excellent overview of the system. While the example can show the encapsulation that is achieved with layered architectures, it is too simple to be considered a typical software-intense system. In our opinion, the full potential of taking layered architecture to the level of systems architecting can only be exploited if multiple software-intense subsystems are designed according to the same layered architecture. We think that the presented approach is applicable to such systems.

In the presented approach, functional architectures and layered architectures evolve in mutual dependency. In order achieve an appropriate architecture within short time, fast iterations are important due to the corresponding interdependencies. Automatic creation of functional architectures by means of the FAS plugin for various modeling tools

[FAS11a] facilitates fast iterations, because the functional architecture can be re-synthesized with the plugin whenever the grouping of functions has been altered.

While the modeling effort to benefit ratio of layered models seems to be rather high even though they provide an excellent overview, we can imagine that appropriate tool support for combined handling of functional architectures and layered architectures could provide more relative weight on the benefit side. No matter whether the approach is indeed chosen for system development, we expect that the shown procedure is followed at least implicitly, as soon as at least one subsystem has a layered architecture. The question then is which results of the mental process to model explicitly.

So far, our procedure has been described by means of one example, but no generally applicable method for designing layered architectures has been provided. The latter could be a subject of future work.

8 Acknowledgments

The authors like to thank Bernafon Architecture Coordination Group for their substantial contribution to the approach we have presented in this paper and Alexander Lohberg from GfSE's functional architectures working group for his feedback that went into this paper.

References

[FAS12a] www.fas-method.org, browsed Jan. 10th, 2012.

[FAS12b] http://sourceforge.net/projects/fas4md/, browsed Jan. 10th, 2012.

[FAS12c] FAS Plugin for MagicDraw®, User Guide, January 2012.

[KLW11] Korff, A.; Lamm, J. G. and Weilkiens, T.: Werkzeuge für den Schmied funktionaler Architekturen. In Maurer, M. and Schulze, S.O. (eds.), *Tag des Systems Engineering, Hamburg, 9.-11. November 2011*, pp. 3–12. Carl Hanser Verlag, München, Germany. English translation available via www.fas-method.org.

[LEB09] Lamm, J. G.; Espinoza, A. and Berg, A. K. System tests for reconfigurable signal processing systems. TuZ 2009: 21. Workshop für Testmethoden und Zuverlässigkeit von Schaltungen und Systemen. Bremen, Germany, February 2009.

[LW10] Lamm, J. G. and Weilkiens, T. Funktionale Architekturen in SysML. In Maurer, M. and Schulze, S.O. (eds.), *Tag des Systems Engineering, München Freising, 10.-12. November 2010*, pp. 109–118. Carl Hanser Verlag, München, Deutschland. English translation available via www.fas-method.org.

[Mai06] Maier, M.W. System and Software Architecture Reconciliation. Systems Engineering 9(2), S.146–159, 2006.

[MR02] Maier, M. W.; Rechtin, E.: The Art of Systems Architecting. CRC Press, 2002.

[OMG10] Object Management Group (OMG): OMG Systems Modeling Language (OMG SysML™) Version 1.2. OMG Document Number formal/2010-06-01, 2010.