This is an English translation of a German conference paper presented at the 2010 conference of the German "Gesellschaft für Systems Engineering e.V.". Please cite this document as follows: "Lamm, J. G. and Weilkiens, T., Funktionale Architekturen in SysML. In M. Maurer and S.-O. Schulze (eds.), Tag des Systems Engineering 2010, pp. 109–118. Carl Hanser Verlag, München, Germany, November 2010. English translation by J. Lamm."

Functional Architectures in SysML

Jesko G. Lamm[1], Tim Weilkiens[2]

[1] Bernafon AG, Morgenstrasse 131, 3018 Bern, Switzerland
jla <atsign> bernafon.ch

[2] oose Innovative Informatik GmbH, Straßenbahnring 7, 20251 Hamburg, Germany
tim.weilkiens <atsign> oose.de

Abstract: Functional Architectures enable re-use of concepts across multiple generations of technology. This paper shows how to create Functional Architectures in SysML. We give examples of architectural models in SysML resulting from a modeling approach that has been successful in several projects of the authors, amongst others in the hearing instrument domain.

1 Introduction

A function for producing sound was present already in old days' grammophones. Their horn is no longer a state-of-the-art technology, but the related function "Amplify Sound" is still relevant. Modern hearing instruments, for example, provide this function, not with a horn, but rather with microchips and small-sized electroacoustic transducers with dimensions in the order of magnitude of very few millimeters. This example shows: Describing products by their functions will lead to concepts with higher lifetime than will an approach that depends on a certain technology. A functional view also enables a deeper insight into the system [Ack81, Hit07].

In system architecture, this is a reason to use functional architectures. This paper introduces functional architectures and presents a method for creating and modeling them. This is done using [OMG10], the international Standard OMG Systems Modeling Language (SysML), because it has been successfully applied and is supported by currently available modeling tools. All figures in this paper are SysML diagrams.

We have observed the need for function-oriented development of systems, and we like this paper to show how to address it and integrate the solution in a typical Systems Engineering model. We furthermore like to point out how functional architecture can be interlinked with the classic elements of Systems Engineering, like requirements or physical architecture.

This is an English translation of a German conference paper presented at the 2010 conference of the German "Gesellschaft für Systems Engineering e.V.". Please cite this document as follows: "Lamm, J. G. and Weilkiens, T., Funktionale Architekturen in SysML. In M. Maurer and S.-O. Schulze (eds.), Tag des Systems Engineering 2010, pp. 109–118. Carl Hanser Verlag, München, Germany, November 2010. English translation by J. Lamm."

2 Terminology

Whereas physical elements exist in the real world, functional elements are abstract and can only materialize within a model of a system. Functional elements transform modeled entities / quantities like e.g. information, signals, materials, force or energy [Pat82, Ulr95]. Functional elements have different names in literature, for example "functional element" [YW07], "function" [KMN+00] or "system block" [BFN05]. Hence, it is necessary to define terms carefully. The definitions used in this paper have been summarized in table 1.

Functional elements can be decomposed into sub-elements. Similarly, their functions can be decomposed into sub-functions. We call the corresponding activity functional decomposition and its result the function structure. Function structures are hierarchical. Their topmost level of hierarchy depends directly on the user [Spi02] and is closely related to use cases. Not all approaches of functional system description have this topmost level of hierarchy.

The concept of functional decomposition can be found in various pieces of literature [PBFG07 p. 170 and following] [Eis05, p.145-146] [Spi02] [Pat82, p.66; p. 199 and following]. The cited material differs in background. This is why the next section will cover approaches that are compatible with the current context of functional architectures. Where possible, make reference to the corresponding literature.

Term	**Definition**
Function	Input/output relationship [PBFG07, Pat82] of a *functional element*.
Functional element	Abstract system component that defines a relation between at least one input and at least one output by means of a *function*.
Functional block	Model element that is supposed to represent a *functional element* in the model.
Functional unit, functional group, sub-function, sub-element	See the corresponding roles that are assigned to composition relationships in figure 4.
Function structure	The hierarchical structure that results from decomposing *functions* into *sub-functions* and from decomposing *functional elements* into *sub-elements*.
Architecture	Description of the system under development that provides structured views on it by identifying its elements and relating them to each other.
Functional Architecture	*Architecture* based on *functional elements*.

Table 1: Terminology

This is an English translation of a German conference paper presented at the 2010 conference of the German "Gesellschaft für Systems Engineering e.V.". Please cite this document as follows: "Lamm, J. G. and Weilkiens, T., Funktionale Architekturen in SysML. In M. Maurer and S.-O. Schulze (eds.), Tag des Systems Engineering 2010, pp. 109–118. Carl Hanser Verlag, München, Germany, November 2010. English translation by J. Lamm."

3 Method for Creating Functional Architectures

The method describes the steps involved in moving from system requirements to functional architecture. It is independent of the used modeling language. However, dealing with functional architectures requires a language that supports multiple levels of abstraction and different views on the model. We have chosen the SysML language, because it meets these requirements.

3.1 Overview

Figure 1 shows how to proceed: requirements are identified and refined by use cases in SysML, as shown in figure 2. Modeling the activities then adds more details to the use cases [Wei08]. A model of the system's functional architecture now results from grouping activities of the use cases into functional elements.

Note that figure 1 only shows an idealized flow, whereas the practical application of the method will typically come with a different sequence of steps, each of them typically being performed more than once instead of being completed at once.

3.2 Identifying Functional Requirements

Functional requirements about the system under development are the most important input in creating a functional architecture. To identify and specify them, there are different methods, whose choice is independent of the intention of creating a functional architecture. Find a procedure for describing requirements in SysML in [Wei08].

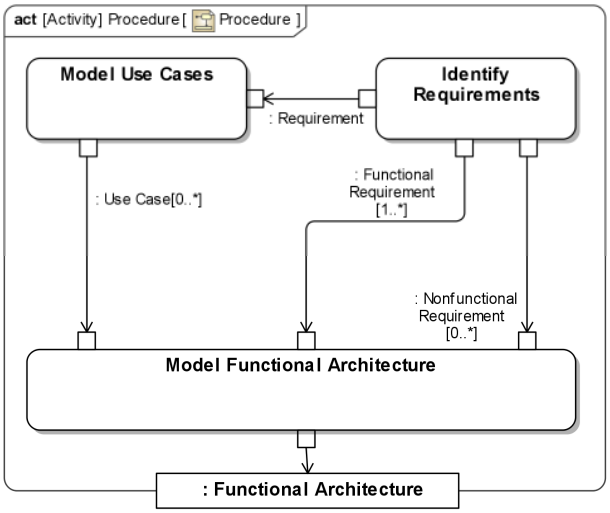

Figure 1: Method overview

This is an English translation of a German conference paper presented at the 2010 conference of the German "Gesellschaft für Systems Engineering e.V.". Please cite this document as follows: "Lamm, J. G. and Weilkiens, T., Funktionale Architekturen in SysML. In M. Maurer and S.-O. Schulze (eds.), Tag des Systems Engineering 2010, pp. 109–118. Carl Hanser Verlag, München, Germany, November 2010. English translation by J. Lamm."

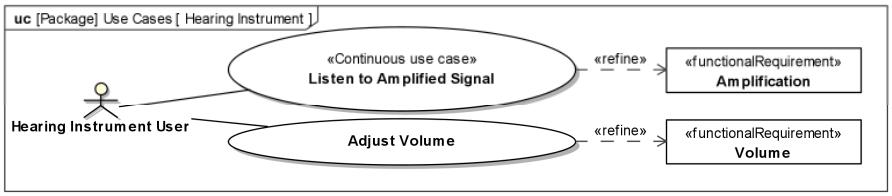

Figure 2: Use case model from the hearing instrument domain

By describing use cases and the flow of activities, the method establishes a transition from requirements like "The system shall provide <xyz>" to functional architecture. Functional requirements are refined by use cases. These provide a view on the functions of the system that is focused on the actors, i.e. those elements that are outside the system, but interact with it. Of course this includes particularly the (human) users of the system, but also external systems. Use cases are a wrapper around the system's functions, defining the preconditions and post-conditions as well as trigger and result. SysML activity diagrams describe the functions.

The control flow between actions of activities is irrelevant for functional architecture. It is only needed in requirements analysis. Most relevant here are the actions themselves and the object flow, which describes input and output objects of the action.

A more detailed description of modeling use cases with activities is provided, for example, by [Wei08].

3.3 Modeling Functional Architecture

The activities of use cases from section 3.2 are a refinement of the functional requirements. They are the key element of the behavioral view on the system. The functional architecture belongs to the static view, which is underlined by the fact that we do not need the control flows of activities. SysML provides a static view on activities that hides the control flow. It is the function tree on a block definition diagram, in which each node represents an activity (=functional element). The tree structure expresses the functions' call hierarchy; that means: a node will be executed in the context of its parent node. This does not necessarily make the function tree a functional decomposition. The SysML composition relationship is the connector in the tree. The roots of function trees are the use cases. They are based on system functions from the actors' point-of-view. A more detailed description on modeling function trees with SysML is provided by [Wei08]. An example is shown in figure 3.

The activities of the function trees can be grouped according to certain criteria. In general, the result has a different structure than the underlying function trees. For the first time, the function structure as a basis of the functional architecture comes into existence.

This is an English translation of a German conference paper presented at the 2010 conference of the German "Gesellschaft für Systems Engineering e.V.". Please cite this document as follows: "Lamm, J. G. and Weilkiens, T., Funktionale Architekturen in SysML. In M. Maurer and S.-O. Schulze (eds.), Tag des Systems Engineering 2010, pp. 109–118. Carl Hanser Verlag, München, Germany, November 2010. English translation by J. Lamm."

In SysML, the function structure is represented in block defintion diagrams. Per functional group, one functional element is modeled as a functional block, i.e. a standard SysML block with a newly defined stereotype "Functional Block". Operations of functional blocks model the actual function, i.e. the input/output relationship between objects that are the block's input and output via its ports. As useful guideline, one can define that each operation of a functional block should match a sub-function of the corresponding functional group. Functional blocks can be described as parts of other functional blocks via the composition relationship. This can be used to model the decomposition of functional elements into sub-elements.

Functional blocks whose functions call each other are connected in the internal block diagram via standard ports and flow ports – the former to model the flow of signals [Ulr95] or information [Pat82], the latter to model flows of material, force or energy [Pat82, Ulr95]. Connections between ports can be inferred partly from the object flows of activity diagrams from section 3.2; in practice, however, internal block diagrams can describe certain matter with a more clear visualization and provide a better overview than activity diagrams – particularly in cases of objects flowing between functional elements resulting from different branches of the function tree (in the hearing instrument example of section 5, we illustrate this with the example of a gain change that is needed in use case "Listen to Amplified Signal" and has its origin in use case "Adjust Volume").

So far, we have described how to proceed. Inspired by [DB04], we summarize the resulting information dependencies in figure 4. In addition to the previously said, the figure shows: non-functional requirements can have an influence on the function structure via architectural decisions. This is particularly relevant in the lower hierarchy levels of the function structure.

4 Heuristics for Obtaining a Sensible Function Structure

It is impossible to group functions according to systematically or even automatically applied criteria [Pat82]. Architecture combines "art" and technology: the creation of functional architectures is an achievement of the system architect and requires thought.

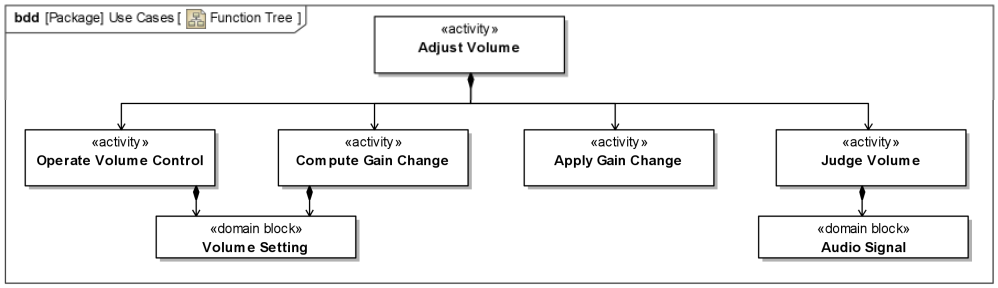

Figure 3: Function Tree

This is an English translation of a German conference paper presented at the 2010 conference of the German "Gesellschaft für Systems Engineering e.V.". Please cite this document as follows: "Lamm, J. G. and Weilkiens, T., Funktionale Architekturen in SysML. In M. Maurer and S.-O. Schulze (eds.), Tag des Systems Engineering 2010, pp. 109–118. Carl Hanser Verlag, München, Germany, November 2010. English translation by J. Lamm."

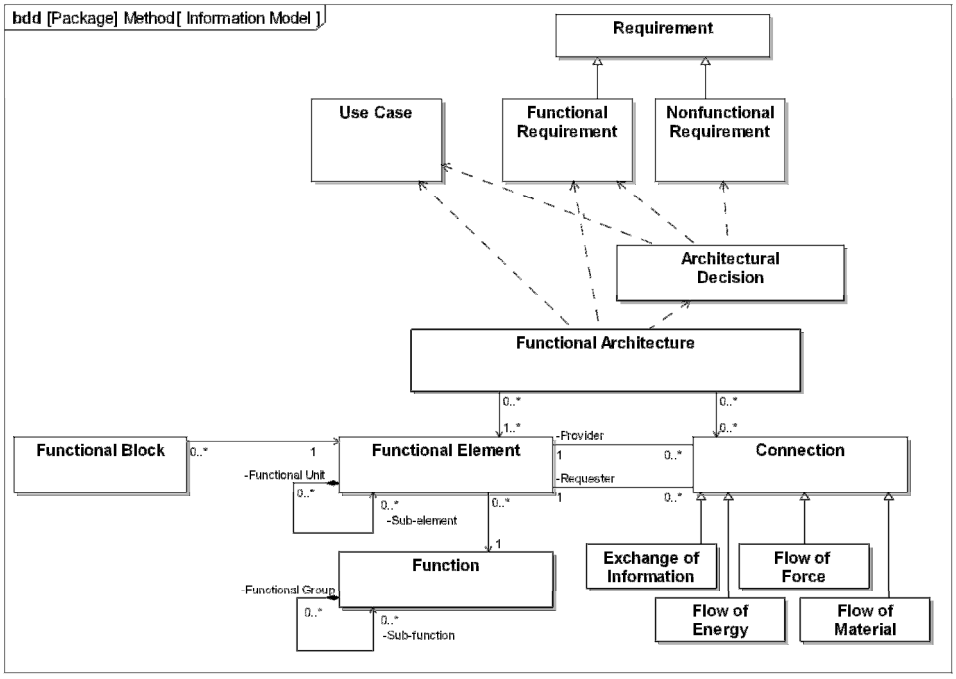

Figure 4: Information Model

Heuristics can support the architect [MR02] if no procedure is proven to be the optimum one, like in the context of this paper. In the following, we provide heuristics that help obtaining a function structure by grouping activities of the function tree. The grouping aims at functional groups that are "as independent as possible; that is […] with low external complexity and high internal complexity" [MR02]. This makes the functional architecture invariant to changes and facilitates deriving an effective physical architecture from it.

Use grouping criteria of existing groups: A system is rarely developed completely from scratch, but it is usually based on an existing system. The outline of existing system documentation from prior art or similar systems can indicate possible ways of grouping functions. Ideally, interviews with the system developers should be made to find out if the grouping was useful in practice. This way, known structures will be created and team members will find them intuitive to use. However, grouping criteria have to be re-assessed with caution: they can be of technical rather than conceptual nature. A grouping based on technical constraints is not desirable, because it will lead to a functional architecture that contains implicit technological decisions, making it more difficult to find alternative solution scenarios.

Abstract and secondary use cases define a functional group: The use case model already reveals potential functional groups. An abstract use case represents commonalities between several concrete use cases. Its functions can be assigned to one functional group, either in order to complete it or in order to leave it open for further

This is an English translation of a German conference paper presented at the 2010 conference of the German "Gesellschaft für Systems Engineering e.V.". Please cite this document as follows: "Lamm, J. G. and Weilkiens, T., Funktionale Architekturen in SysML. In M. Maurer and S.-O. Schulze (eds.), Tag des Systems Engineering 2010, pp. 109–118. Carl Hanser Verlag, München, Germany, November 2010. English translation by J. Lamm."

assignment of functions from the concrete use cases. The modeling of abstract and secondary use cases is covered by [Wei08].

One functional group takes the functions that are related to system actors: Functions having a direct relationship with system actors are part of the system's input/output logic. Often they only have little in common with the actual system functions that do the processing of the inputs and produce the outputs. In that case, they are good candidates for a separate functional group.

Function calls imply cohesion: Functions call other functions, resulting in a network of call relationships. Clusters in that network are potential functional groups. They can be derived easily from function trees. The composition relationships in a function tree can be represented as a matrix. Rows and columns represent the activities, and a marker is placed in a matrix cell if the corresponding row activity calls the corresponding column activity. Rows and columns have to be moved such that markers accumulate in distinct parts of the matrix. The resulting clusters map to row and column activities that can potentially be grouped into a functional group.

Functions that share data can be grouped: It can be assumed that two functions belong to closely related domains if the output of one of them is the other's input. This connection can easily be found in the object flow of the activity diagrams. However there can be an implicit control flow that is irrelevant here. Again, the function trees are most suited for this heuristic: they can show the objects (data) of functions (see figure 3).

5 Example

Figure 5 gives an example by showing the functional architecture of a simplified hearing instrument in a SysML representation. It is based on a much more elaborate functional architecture, which has been re-used throughout multiple hearing instrument projects.

To keep the example simple, it has been assumed that a hearing instrument can only amplify sound and apply volume changes. Already the use cases of figure 2 have been based on this simplification. This ways the functional architecture from figure 5 matches the use cases: The functional element "Adjust Volume" belongs to the use case of same name.

This is an English translation of a German conference paper presented at the 2010 conference of the German "Gesellschaft für Systems Engineering e.V.". Please cite this document as follows: "Lamm, J. G. and Weilkiens, T., Funktionale Architekturen in SysML. In M. Maurer and S.-O. Schulze (eds.), Tag des Systems Engineering 2010, pp. 109–118. Carl Hanser Verlag, München, Germany, November 2010. English translation by J. Lamm."

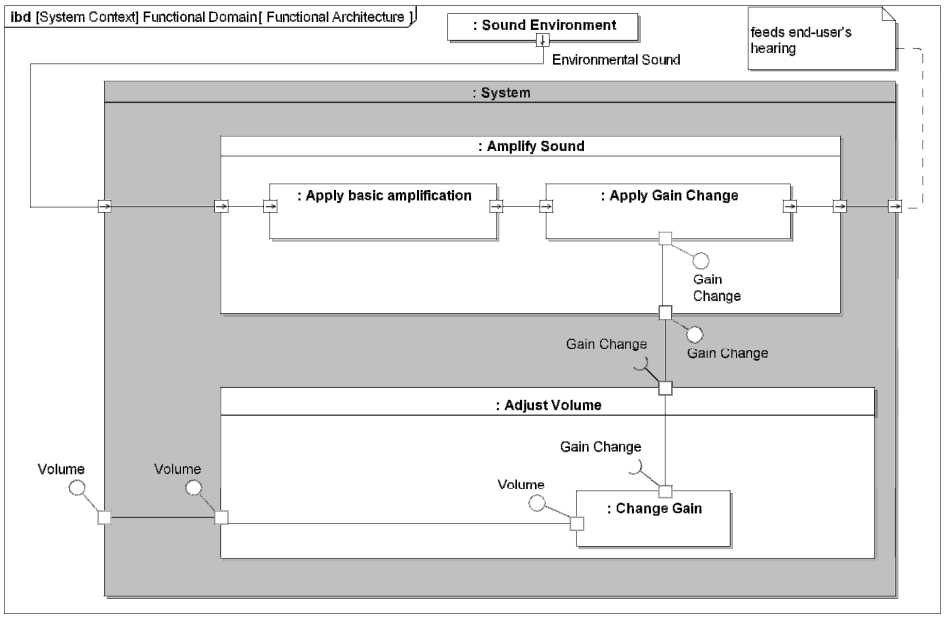

Figure 5: Simplified functional architecture of a sample hearing instrument

The use case "Listen to Amplified Signal" is enabled by the functional element "Amplify Sound", which matches the function of same name that has been mentioned in the introduction. This functional element does not only model amplification of sound, but also the possibility to change gain, which is offered to the outside via an interface and is in the given example beneficial for the hearing instrument user.

6 Moving into Implementation

A functional architecture itself cannot be implemented. Therefore a physical solution providing the identified functions is needed before the architecture can be realized in a system. Again, this solution ideally comes with a structured view, which we call "physical architecture". As an example, we show in figure 6 how a potential physical architecture could look like if it was to realize the functional architecture from figure 5 on the basis of nowadays digital technology according to [PSH+04].

Possible procedures for realizing functions in a physical system have been described by the literature ([Ulr95], [KMN+00], [BFN05], [Hit07], [Pat82]). System architects are mainly interested in the allocation of elements in physical architecture by functional elements ("functional-to-physical mapping" [Hit07]). This can be expressed in SysML by means of an "allocate" relationship. The functional block "Amplify Sound" from figure 5 thus has incoming "allocate" relationships from those physical blocks that work together to provide the function "Amplify Sound". In a typical hearing instrument according to figure 6, this would e.g. be "allocate" relationships coming from the

This is an English translation of a German conference paper presented at the 2010 conference of the German “Gesellschaft für Systems Engineering e.V.”. Please cite this document as follows: “Lamm, J. G. and Weilkiens, T., Funktionale Architekturen in SysML. In M. Maurer and S.-O. Schulze (eds.), Tag des Systems Engineering 2010, pp. 109–118. Carl Hanser Verlag, München, Germany, November 2010. English translation by J. Lamm.“

elements “Microphone”, “Input Stage”, “Interconnection Network”, “ Filter 1”, “Filter 2”, “Signal Processor”, “Output Stag” and “Loudspeaker”.

One functional architecture can map to different physical architectures. Figure 7 illustrates this in showing an alternative physical architecture of the sample hearing instrument: it is based on outdated analog technology, but still matches the functional architecture. The latter one thus does not only describe a modern device according to figure 6, but also analog hearing instruments of the previous millennium. The example shows: functional architectures can stay valid across multiple generations of technology.

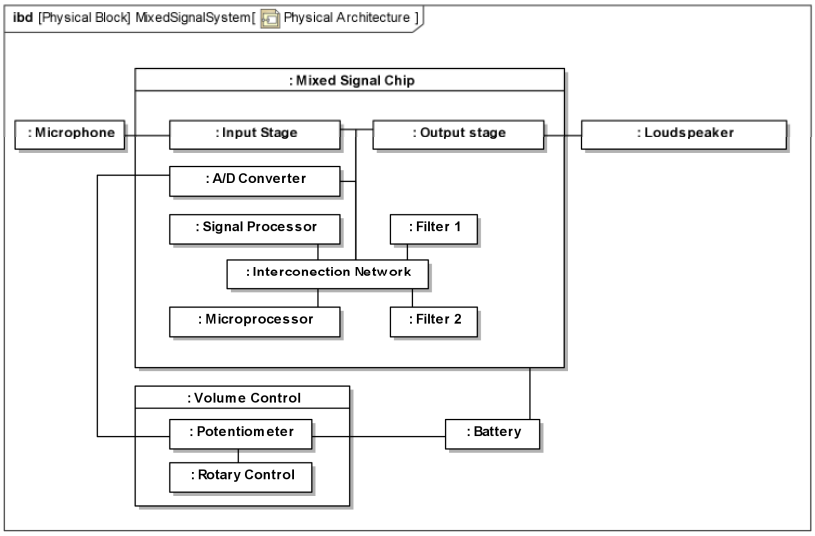

Figure 6: Architecture Alternative of a Physical Architecture for implementing the sample hearing instrument in modern digital technology, based on [PSH+04]

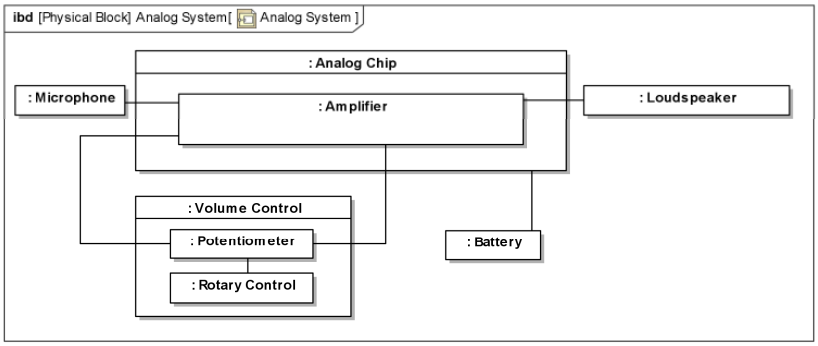

Figure 7: Architecture Alternative of Physical Architecture based on Analog Technology

This is an English translation of a German conference paper presented at the 2010 conference of the German "Gesellschaft für Systems Engineering e.V.". Please cite this document as follows: "Lamm, J. G. and Weilkiens, T., Funktionale Architekturen in SysML. In M. Maurer and S.-O. Schulze (eds.), Tag des Systems Engineering 2010, pp. 109–118. Carl Hanser Verlag, München, Germany, November 2010. English translation by J. Lamm."

7 Conclusion

We have shown an approach for deriving functional architectures from requirements and use cases and modeling them in SysML. Using functional architecture will lead to re-usable models, which can support multiple generations of technology. We have exemplified this in the domain of hearing instrument, with a simplified functional architecture, which is valid for both systems based on outdated analog technology and the ones using nowadays technology. While the functional elements of the system have a long life cycle, the physical architecture of the system will be developed further with the evolution of technology and will over time improve the implementation of the functions the user is interested in.

Acknowledgements

We like to thank Mr. Matthias Dänzer, Bernafon AG, as well as the TdSE 2010 reviewers for their feedback and *No Magic, Inc.* for supporting this paper with additional licenses of the modeling tool.

References

[Ack81] Ackoff, R. L.: Creating the Corporate Future: Plan or be Planned For. Wiley, 1981.

[BFN05] Blume, H.; Feldkaemper, H. T.; Noll, T. G.: Model-based exploration of the design space for heterogeneous systems on chip. Journal of VLSI Signal Processing, 40:19-34, 2005.

[DB04] Daniels, J.; Bahill, T.: The hybrid process that combines traditional requirements and use cases. Systems Engineering, 7(4):303–319, 2004.

[Eis05] Eisner, H.: Managing complex systems. Wiley, 2005.

[Hit07] Hitchins, D. K.: Systems Engineering. John Wiley & Sons, 2007.

[KMN+00] Keutzer, K.; Malik, S.; Newton, A. R.; Rabaey, J. M.; Sangiovanni-Vincentelli, A.: System-level design: Orthogonalization of concerns and platform-based design. IEEE Trans. Computer-aided Design of Integr. Circuits and Syst., 19(12):1523-1543, 2000.

[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.

[Pat82] Patzak, G.: Systemtechnik – Planung komplexer innovativer Systeme. Springer, 1982.

[PBFG07] Pahl, G.; Beitz, W.; Feldhusen, J.; Grote, K.-H.: Engineering Design. Springer, 2007.

[PSH+04] Paker, Ö.; Sparsø, J.; Haandbæk, N.; Isager, M.; Nielsen, L. S.: A Low-Power Heterogeneous Multiprocessor Architecture for Audio Signal Processing. Journal of VLSI Signal Processing, 37: 95-110, 2004.

[Spi02] Spielberg, D. E.: Methodik zur Konzeptfindung basierend auf technischen Kompetenzen. Dissertation, RWTH Aachen, Shaker-Verlag, 2002.

[Ulr95] Ulrich, K.: The role of product architecture in the manufacturing firm. Research Policy, 24:419 – 440, 1995.

[Wei08] Weilkiens, T.: Systems Engineering mit SysML / UML. dpunkt.verlag, 2008.

[YW07] Yassine, A. A.; Wissmann L. A.: The Implications of Product Architecture on the Firm, Systems Engineering, 10(2):118-137, 2007.