J. Michael, J. Pfeiffer, A. Wortmann (Hrsg.): Modellierung 2022 Satellite Events,
Digital Library, Gesellschaft für Informatik e.V. 106

Modeling an Anomaly Detection System with SpesML

An Experience Report

Maximilian Junker[1], Henning Femmer[2]

Abstract: SpesML is an instantiation of the SPES methodology for cyber physical systems using SysML. However, SpesML is still under development and urgently requires evaluation with practical examples. This experience report describes our study of using SpesML for an anomaly detection system. The goals of the case study are to evaluate feasibility, benefits, and shortcomings of both the tool and the methodology iteratively at early stages of the project. The results are already promising with respect to both methodology and tool; however, the work continuously identifies suggestions for adaptations and future work regarding both.

Keywords: Model-based Systems Engineering (MBSE), Iterative Development, SPES, SpesML

1 Introduction

Many companies are currently discussing introducing Model-based Systems Engineering (MBSE) into their processes. However, adopting MBSE in the industrial practice still comes with considerable effort to define, implement, and customize a suitable MBSE method and tooling.

The SPES series of projects are a joint effort by industry and academia to provide a methodology for MBSE to ease introduction [Br12]. Its primary aim is to base MBSE on a precise system model. Previous project succeeded in defining this methodology.

In the SpesML project the additional aim is to port SPES to SysML, a standardized notation that is picking up dissemination in industry, and to provide a tool that supports the SPES method. Part of such an endeavor is then, of course, to validate the developed methods and tools in exemplary cases.

In this work, we present the case of developing an anomaly detection system using the SpesML methodology. Due to the early stage of both methodology and research, this paper qualitatively evaluates the feasibility of the methodology, as well as the expressiveness and usability of the tooling.

[1] Qualicen GmbH, Rosa-Bavarese-Straße 15, 80639 München, Germany, maximilian.junker@qualicen.de
[2] FH Südwestfalen University of Applied Sciences, Haldener Straße 182, D-58095 Hagen, Germany, femmer.henning@fh-swf.de

In the following, in Sec. 2 first look at background and related work. Afterwards, in Sec. 3, we describe the case study and the different iterations, in which we created the model. We then describe the resulting model in Sec. 4. Finally, we discuss these results w.r.t. to the aforementioned research questions in Sec. 5, and summarize the presented work in Sec. 6.

2 Background and Related Work

This work is based on the MBSE frameworks SPES and SpesML. SpesML itself is based on SysML, for which few methodologies exist. In the following, we summarize these and provide pointers for further reading.

2.1 SysML and SysML Methodologies

SysML is a standardized modeling language for systems engineering. It is originally based on UML but extends UML to better match the needs for modeling systems instead of software. Please refer to [FMS14] for a detailed introduction into SysML [OM19]. SysML however, is a language, not a methodology. It therefore only provides the building blocks and endless possibilities how to apply the language.

Various approaches have tried to fill this gap. In no particular order, the most prominent approaches probably are Dassault's own MagicGrid [Mo20], the ARCADIA approach [Ro16; Ro17] strongly tied to the eclipse capella tool[3], and the SYSMOD methodology [We16].

2.2 SPES & SpesML

For simplicity, we do not want to explain SPES in all detail here. Please refer to any of the published material for fundamentals [Bö14a; Br12], extensions [Bö21; Po16], case studies [Bö14b] or introduction methodology [We21] for this. Instead, we just provide a very rough overview in this chapter.

SPES is a framework for MBSE. It defines a set of models to describe different aspects of a system under development with varying level of detail. To this end, SPES defines four core viewpoints:

- Requirements Viewpoint: Contains the requirements to the system.
- Functional Architecture Viewpoint: Contains the system function of the system and breaks those down into whitebox functions.

[3] https://www.eclipse.org/capella/

- Logical Architecture Viewpoint: Contains a component architecture of the system which is independent from the technical realization.
- Technical Architecture Viewpoint: Contains an architecture of technical, discipline-specific components (e.g., mechanical components and software components)

Additionally SPES defines the concept of layers of granularity which allows to identify subsystems and develop those independently. Finally, all models in SPES are based on a common universal interface model, a common system model, and an overarching architecture model providing the base for refinement and tracing across viewpoints and levels of granularity.

While SPES is independent of a specific tool and modeling language, SpesML is an instantiation of SPES using SysML and the commercial modeling tool Cameo Systems Modeler[4] by 3ds Dassault Systemes. The SpesML workbench is a plugin to Cameo providing SPES concepts as well as advanced analyses and simulation in Cameo.

2.3 Research Gap

Currently, there are no published experience reports for applying SPES to SysML with SpesML. This work addresses this gap.

3 Study Setup and Execution

In this chapter, we describe the goals of the study, the study object, and how the study was executed.

3.1 Goals of the Case Study

At this early stage, the goal of this case study is to identify potential for improvement in the method as well as in the tool. In consequence, the goals of the case study are two-fold:

- First, the case study shall evaluate the applicability of the SPES method and potentially serve as a blueprint for creating MBSE models based on SPES.
- Second, the case study shall evaluate the expressiveness and usability of the SpesML tooling.

[4] https://www.3ds.com/products-services/catia/products/no-magic/cameo-systems-modeler/

3.2 Study Object: Anomaly Detection

The System under Development is an anomaly detection system (ADS). The ADS example is taken from the following real world problem: When sensors fail, it is often not obvious that they are leaving a nominal operation mode, e.g., but not replying anymore at all. Instead, the sensors continue providing data, but just *incorrect* data. This is usually recognized by irregularities (i.e., anomalies) in the provided sensor data. The main task of an ADS system, therefore, is to use sensor data originating from a monitored system (e.g., a machine) to detect anomalies which could potentially lead to damage of that system. The ADS works in two phases: In the first phase it monitors the sensor data while the monitored system works nominal. From the data gathered in this way, the ADS creates a benchmark. In the second phase, the ADS uses this benchmark to asses the sensor data and determine if there is an anomaly.

3.3 Study Subjects

The model was created by the two authors of this work, as well as two further employees of Qualicen GmbH. Three modelers are SysML and SPES experts, whereas for one modeler it was his first expose to SPES, but not his first to SysML.

3.4 Case Study Execution

The case study was conducted during the course of the research project. The tooling used was developed in parallel with the study execution and the development of the model. Accordingly, the model was created in several iterations. In each iteration we changed the model according to changes in the method as well as integrated new aspects.

During the course of the modeling, we documented findings regarding the applicability of the method, possible improvements, and workarounds. However, we did not perform any specific analysis to uncover problems, but instead used an opportunistic approach. Hence, the list of issues cannot be considered complete.

Iteration 1: Scope and high-level requirements In the first iteration, we defined the scope of the system under development and developed an initial set of high-level stakeholder requirements. At this very early stage of the research project, there was only preliminary tool support. We nevertheless created this scope document and the requirements in the tool as Cameo supports these artifacts out of the box and we could later make adoptions as the SpesML tool made progress.

Iteration 2: Initial logical architecture In the second iteration, we would have ideally designed the functional viewpoint. However, since the logical viewpoint of SpesML was finished earlier, we decided to create a prototype of the logical architecture ahead, which we would later refine. Therefore, we created an initial logical architecture in the second iteration, where we defined coarse components, e.g., for data processing, storage, and interfacing.

Iteration 3: Requirements Refinement & Functional Architecture In a third iteration, we refined the initial coarse grained stakeholder requirements with a larger set of fine-grained system requirements. These included functional requirements as well as quality requirements and design constraints. From the functional requirements we developed a functional black-box model, containing the system functions located at the system interface. We then refined these black-box functions by giving a functional white-box model for each system function. We added trace-links connecting the different requirements levels as well as connecting the requirements to the functional architecture.

Iteration 4: Refined logical architecture and simulation In the fourth iteration, we refined the logical architecture to faithfully realize the functional architecture and the requirements. We further added behavior descriptions to logical components in order to be able to simulate the components.

4 Resulting Model

In the following, we show the models developed in the requirements-, logical-, and functional viewpoint. Although there is a preliminary technical viewpoint available, we will not go into details here, since at the time of writing, the tool support and the method for the technical viewpoint has not been completed yet.

4.1 Requirements Viewpoint

In the requirements viewpoint we have three main artifacts: (1) the scope document, (2) the stakeholder requirements, and (3) the refined system requirements. For the scope document we used the Free Forms Diagram, which is provided by Cameo and which supports creating informal documents containing text and images. The document describes the ADS and provides background information. For the stakeholder requirements and the system requirements, we created a SpesML Requirements Package as well as a SpesML Requirements Table. An excerpt can be found in Fig. 1. It shows the a subset of the functional requirements, focussing on reporting aspects (i.e. querying the sensor data by various variables). The only further requirements attributes are status, text, and type.

#	Name	Status	Text	Requirement Type
1	System Requirements			
2	Functional Requirements			
3	Reporting			
4	Query Sensor Data by Time	reviewed	When the system receives a sensor query specifying a time range, the system returns sensor data with timestamps in that range.	Functional
5	Time Data from Database	proposed	The system checks time range from database.	Functional
6	Data with Timestamps	proposed	The system returns sensor data with timestamps according to database time range.	Functional
7	Query Sensor Data by Source	reviewed	When the system receives a sensor query specifying data sources, the systems returns sensor data that originate from that source.	Functional
8	Specified Source Identification	proposed	The system identifies the specified source and retrieves the source's sensor data from database.	Functional
9	Sensor Data from Source	proposed	The system returns the sensor data originating from the specified source.	Functional
10	Query Sensor Data by Value	reviewed	When the system receives a sensor query specifying a value range , the system returns sensor data with values in that range.	Functional
11	Range of Values from Database	proposed	The system retrieves all sensor data values withing the specidifed value range from database.	Functional
12	Sensor Data by Value	proposed	The system returns all values withing the specified value range.	Functional
13	Query Sensor Data by Anomaly	reviewed	When the system receives a sensor query specifying an annomaly id, the systems returns all sensor data which contributed to the anomaly.	Functional

Fig. 1: An excerpt of the system requirements table.

4.2 Functional Viewpoint

For the functional viewpoint we first created a functional black-box model which contains the system functions (i.e., the functions located at the system boundary). For ADS, the black-box model consists of three system functions (see Fig. 2):

1. **Manage Benchmarks** to create and manage benchmarks of sensor data,
2. **Detect Anomalies** to monitor a system for anomalies during regular application, and
3. **Reporting**, to allow to retrieve historical sensor data and warnings based on a query.

Additionally, the method allows to have communication between black-box function, when the communication relates to a mode of the system. In our case, we used such mode channels to model the communication of created sensor benchmarks and warnings.

For each system function we created a whitebox model detailing how a system function is realized by a network of communicating whitebox functions. Fig. 3 shows an example of the whitebox model for the system function *Reporting*. In this case the whitebox model consists of eight whitebox function describing the internals of the black box system function. Note that the interface seen from the black-box perspective is the same as seen from the whitebox perspective.

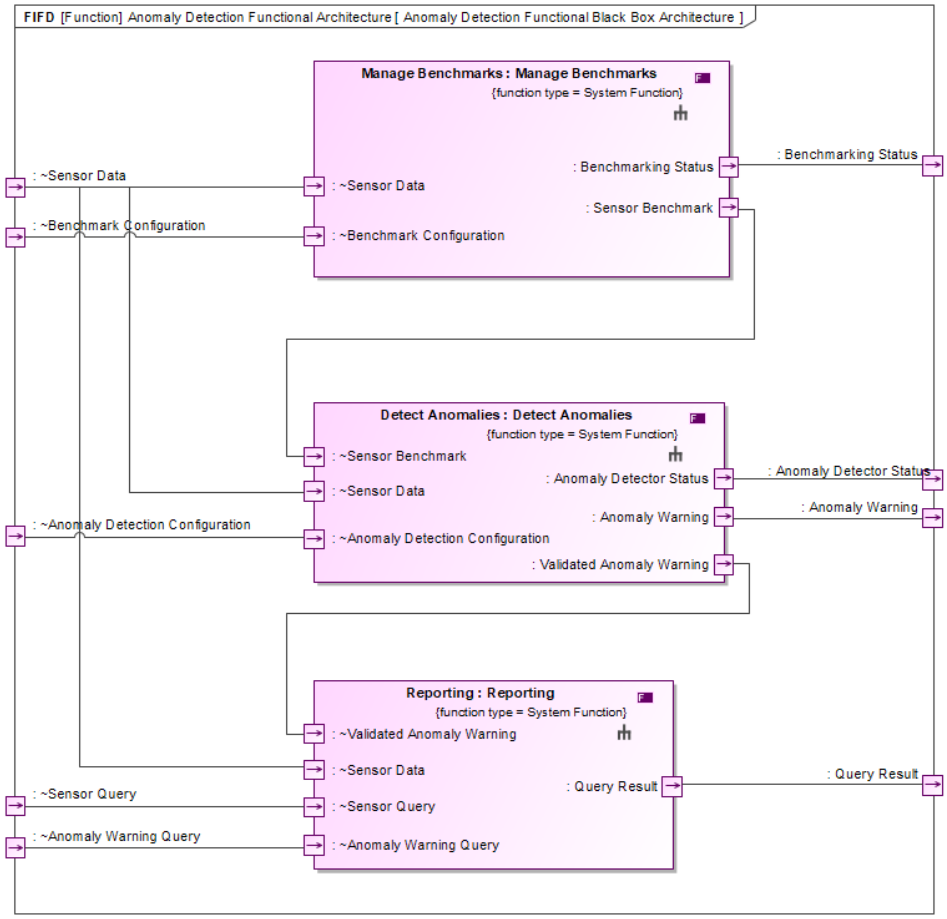

Fig. 2: Functional Black-Box Architecture of the Anomaly Detection

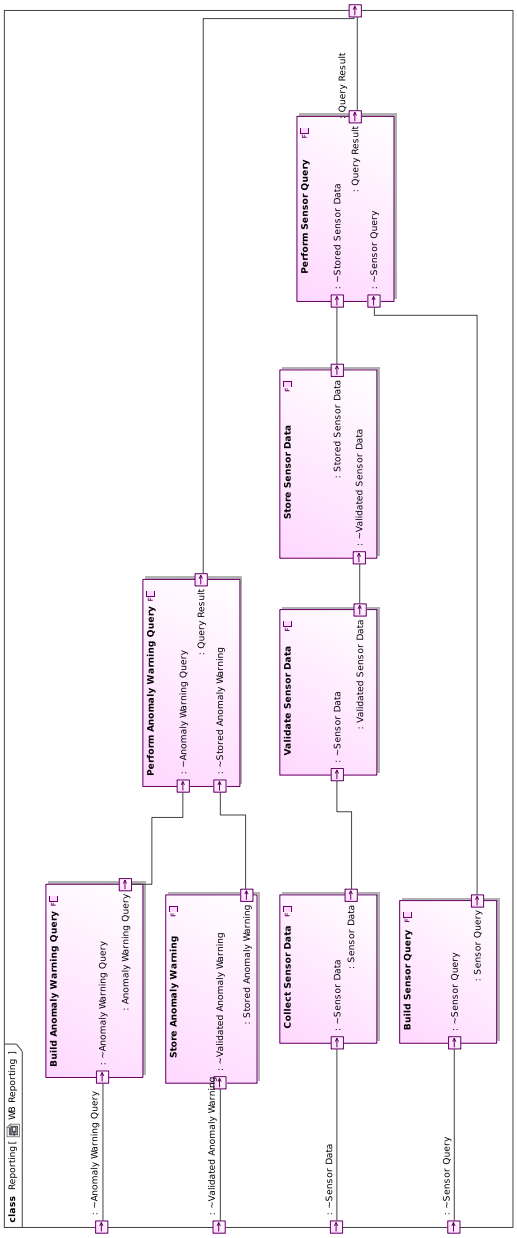

Fig. 3: Whitebox model of the system function Reporting

4.3 Logical Viewpoint

The logical viewpoint of the ADS describes how the functionality described by the functional viewpoint can be realized in a consistent architecture. Where in the functional architecture duplicate functionality exists, e.g., for preprocessing, this is resolved in the logical architecture (for example by a central data processing component). We created a logical architecture (see Fig. 4 with several decomposition levels (not to confuse with layers of granularity). Most components on the highest level were further broken down into further logical components.

Just when we could not break down a component any further, we modeled the behavior of the component using a state machine. Fig. 5 shows the state machine of the model controller. This component controls wether the system is recording a benchmark, performing monitoring or none of those.

4.4 Tracing

Apart from creating the artifacts outlined above, we established tracing relationship between different artifacts. Specifically, we created the following types of trace links

- From stakeholder requirements to system requirements
- From quality system requirements to functional system requirements, when a quality requirement (e.g., security) is realized through a system function or a functional requirement to a system function
- From functional requirements to black-box functions and whitebox functions
- From requirements to logical or technical components, in case a requirement is not realized by a functionality.
- From whitebox functions to logical components

Figure 6 shows an extract of the trace links between the system requirements and the functional architecture.

5 Discussion: Feasibility and Findings

The goal of the case study was, first, to evaluate the applicability of the SPES method, and, second, to evaluate the expressiveness and usability of the SpesML tooling.

Overall, we could so far successfully model the ADS. However, we found the following issues: During the modeling and project internal review rounds we gathered issues regarding the method and the current tooling. Below we report a selection of these issues.

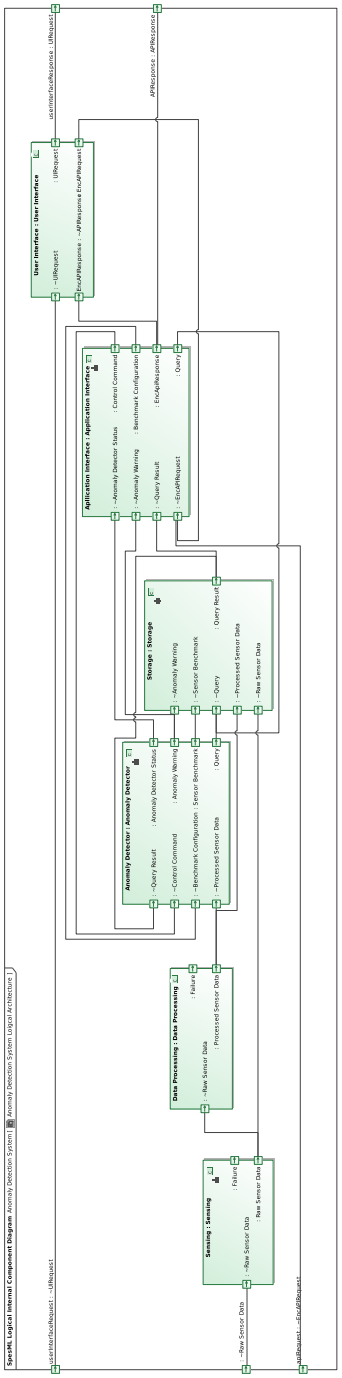

Fig. 4: Logical Architecture of the Anomaly Detection

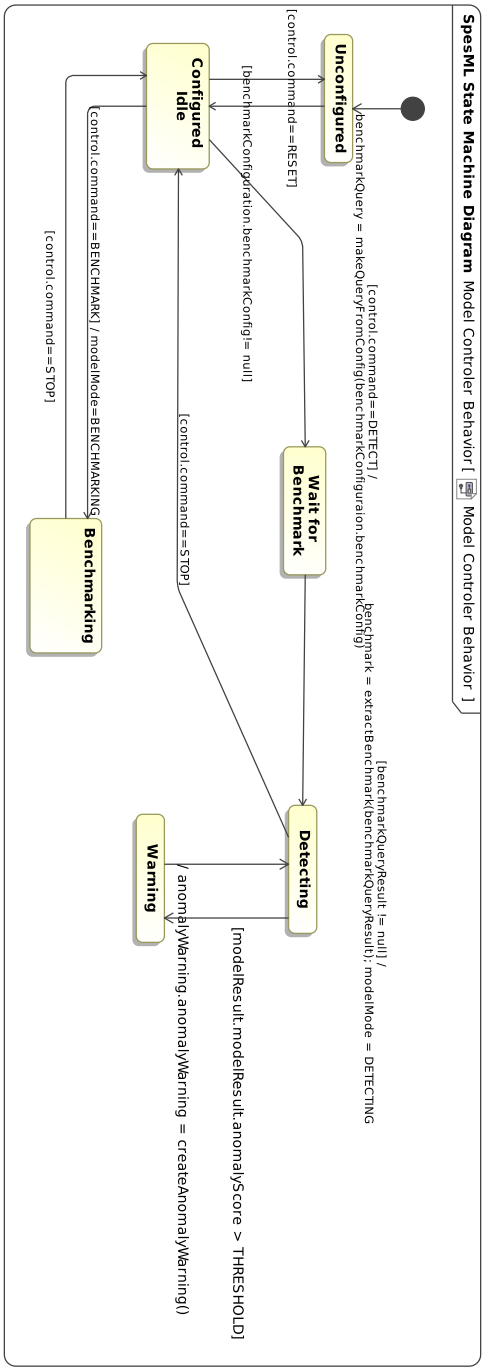

Fig. 5: State Machine of the Model Controller component

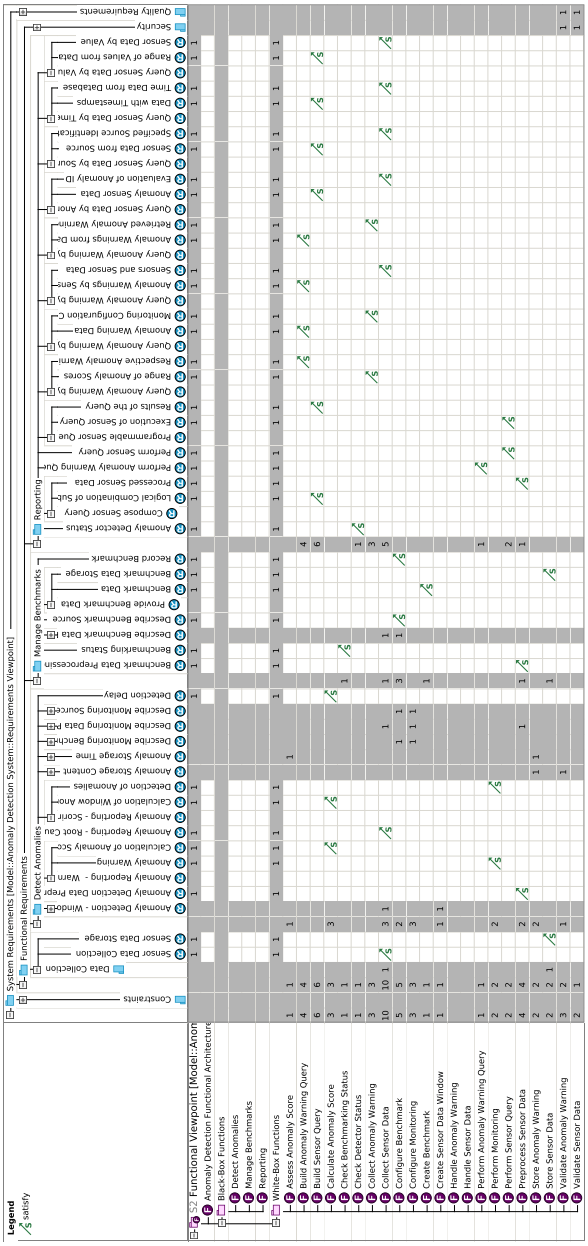

Fig. 6: Excerpt of the tracing links between system requirements and functional architecture

- Requirements: Currently there is the possibility to categorize requirements, link requirements to realizing model elements, and link requirements to each other. However, as pointed out in a review round, often requirements originate from the architecture work. Therefore, it would be helpful to link requirements to the design decision from which they originated.
- Tracing: Currently it is possible to trace requirements to whole functions. It is also possible to trace functions to logical components. However, especially in case of large interfaces, it would be helpful to create traces on a more fine granular level, e.g., between requirements and ports, for example to ease requirements verification and validation.
- Compatible ports: In order to be compliant with the underlying formal universal interface model, there are strict rules regarding the compatibility of ports. However, this leads to an inflexibility regarding the connection between ports and in general to a large number of ports.
- Behavior modeling: Currently, there are specific rules regarding the formulation of guards and effects in state machines, however advanced tool support (e.g., autocompletion in guards) is missing. This would ease the formulation of valid behavior models. Furthermore, currently only state machines can be used to model behavior. Certain types of behavior, e.g., data processing, are not naturally modeled with state machines. In this case, other types of behavior models could be beneficial.

W.r.t. the aforementioned goals, we could see the behavior modeling aspect as a future issue in expressiveness. However, since it did not impact us in our study, we suggest to analyze this issue in future work. For our own system under development, we did not identify gaps in expressiveness, neither in the tooling nor the methodology. We did however, identify a set of potential issues which could improve the usability in future work.

6 Summary

In this work, we gave an experience report on a case study that we conducted in the context of MBSE. With this report, we demonstrated the current state of the SPES modeling method and the SpesML modeling workbench based on Cameo Systems Modeler. We showed, at the example of an Anomaly Detection System, the models in the requirements-, functional-, and logical viewpoint. The focus of the execution of the study, however, was the analysis of the methodology and tooling against expressiveness and usability. The results show that we were able to model the ADS using the SpesML methodology and tooling. However, during the modeling, we identified four suggestions for adjustments.

The presented work was heavily influenced by the current status and the incremental improvement of the tooling during the project, as well as by the inside knowledge and experience of the modelers. Future work will therefore (1) extend the model in the technical

viewpoint and execute simulations, (2) derive hypotheses on the quantitative and qualitative improvements through the method, (3) conduct further studies on different study objects and with other study subjects, in particular non-insiders as modelers, and finally (4) quantify the feedback and analysis provided based on the derived hypotheses.

7 Acknowledgements

We want to thank Alexander Knerr for his contributions during creation of the model. This work was funded by the German Federal Ministry of Education and Research (BMBF) under grant no. 01IS20092K.

Literatur

[Bö14a] Böhm, W.; Henkler, S.; Houdek, F.; Vogelsang, A.; Weyer, T.: Bridging the gap between systems and software engineering by using the SPES modeling framework as a general systems engineering philosophy. Procedia Computer Science 28/, S. 187–194, 2014.

[Bö14b] Böhm, W.; Junker, M.; Vogelsang, A.; Teufl, S.; Pinger, R.; Rahn, K.: A formal systems engineering approach in practice: An experience report. In: Proceedings of the 1st International Workshop on Software Engineering Research and Industrial Practices. S. 34–41, 2014.

[Bö21] Böhm, W.; Broy, M.; Klein, C.; Pohl, K.; Rumpe, B.; Schröck, S.: Model-Based Engineering of Collaborative Embedded Systems: Extensions of the SPES Methodology. Springer Nature, 2021.

[Br12] Broy, M.; Damm, W.; Henkler, S.; Pohl, K.; Vogelsang, A.; Weyer, T.: Introduction to the SPES modeling framework. In: Model-Based Engineering of Embedded Systems. Springer, S. 31–49, 2012.

[FMS14] Friedenthal, S.; Moore, A.; Steiner, R.: A practical guide to SysML: the systems modeling language. Morgan Kaufmann, 2014.

[Mo20] Morkevicius, A.; Aleksandraviciene, A.; Armonas, A.; Fanmuy, G.: Towards a common systems engineering methodology to cover a complete system development process. In: INCOSE International Symposium. Bd. 30. 1, Wiley Online Library, S. 138–152, 2020.

[OM19] OMG: OMG Systems Modeling Language (OMG SysML), Version 1.6, Object Management Group, 2019, URL: http://www.omg.org/spec/SysML/1.6/.

[Po16] Pohl, K.; Broy, M.; Daembkes, H.; Hönninger, H.: Advanced model-based engineering of embedded systems. In: Advanced Model-Based Engineering of Embedded Systems. Springer, S. 3–9, 2016.

[Ro16] Roques, P.: MBSE with the ARCADIA Method and the Capella Tool. In: 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016). 2016.

[Ro17] Roques, P.: Systems architecture modeling with the Arcadia method: a practical guide to Capella. Elsevier, 2017.

[We16] Weilkiens, T.: SYSMOD-The systems modeling toolbox-pragmatic MBSE with SysML. http://model-based-systems-engineering.com, 2016.

[We21] Weyer, T.; Goger, M.; Koch, W.; Kremer, B.: Implementation Strategy for Seamless Model-based Systems Engineering. ATZ worldwide 123/7, S. 66–71, 2021.