Integrating a Signaling Component Model into a Railway Simulation

Daniel Schwencke[1]

Abstract: The validation of software component models is an important activity: The software product usually is derived more directly from a model than from a classical textual specification. Modeling tool suites support this activity by providing code generation and model simulation features. For simulation, an environment for the component needs to be set up: This may be stubs, or the real environment. We report on a third approach where a signaling component model is integrated into a railway simulation. We present the steps taken and what needs to be considered, the integration architecture, and give an account of which kinds of problems have been detected thanks to the simulation. From the successful integration we conclude that the generic model as well as the configuration data used for the integration are valid. Finally, we compare the aforementioned approaches to environments for model simulation.

Keywords: model; validation; simulation; railway; signaling; Radio Block Center; RBC; ETCS

1 Introduction

Model-based approaches have gained visibility over the last years. They are promising both in terms of higher product quality — due to the precision and compactness of models which can help to cope with system complexity — and greater efficiency — due to applicability of automated tools which verify properties or transform the model. Considering also the initial modeling effort, the latter in particular pays off when changes need to be made to a system as part of a development cycle or maintenance; they only must be made to the model instead of different documents and artifacts.

Models can be used for different purposes and at different stages of a system life-cycle. Here we are interested in using models in the context of system validation. For example, models can be used for system specification, and this specification can then be executed in order to validate it. Thus the model allows for testing already in this early life cycle-phase; in case of standardized specifications even independent from a concrete realization project. Models can also be used as reference or for test case generation ("model based testing") in order to validate real system implementations.

In some application areas model-based approaches have spread widely. For example in the automotive sector it seems a convenient way to support variant management. In the current

[1] German Aerospace Center (DLR), Institute of Transportation Systems, Lilienthalplatz 7, 38108 Braunschweig, Germany; daniel.schwencke@dlr.de

paper we present an example from the railway domain, where their adoption seems to be significantly lower. Besides being generally more safety-critical and conservative, other reasons for that may be the generic nature of railway signaling components (i.e. they need to be configurable for arbitrary track layouts) and (anticipated) risks for the mandatory component certification.

Nevertheless, Deutsche Bahn (DB) has produced a set of interface specifications for their new "digital interlockings"using models (NeuPro project) in the Systems Modeling Language (SysML) and has started to carry that approach to a European level (EULYNX, see [EUL18]). Starting with the particular interface to the so-called Radio Block Center (RBC) component, we have created an RBC SysML model using PTC Integrity Modeler. We have been reporting on that activity in the article [SHC17] of the previous MBEES workshop. In the current paper, we share our experiences with integrating and running the RBC model as part of a railway simulation in order to validate it. Note that this differs from the validation efforts of DB who validate single state charts through expert users which are provided with a graphical user interfaces for that state chart.

The paper is organized as follows: Sect. 2 recalls basic information on the RBC model used in our case study, and Sect. 3 gives an overview of the working steps necessary in order to integrate it into the railway simulation. Sect. 4, 5, 6 and 7 describe conditions on this undertaking exported by the different working step activities according to our experience. Sect. 8 and 9 contain information on running the model as part of the simulation and on its validation. Sect. 10 discusses alternatives to using a railway simulation environment, and Sect. 11 summarizes the paper and points to remaining work.

The author would like to thank his colleagues Mirko Caspar and Jonas Grosse-Holz for RBC integration support on the railway simulation side.

2 The Radio Block Center Model

The signaling component chosen for modeling and integration into a railway simulation is the RBC. Being part of the European Train Control System (ETCS), an RBC provides a radio interface between interlocking and train (see Fig. 1). It receives information on the current state of signals and switches from the interlocking as well as on the current position from the train. Based on that information plus data on the railway infrastructure in its area, its main task is to issue so-called "movement authorities"to the train, telling how far and how fast the train may go. Other functionalities include for example emergency stopping of trains or managing temporary speed restrictions. An RBC is a safety critical system and needs to adhere to the highest standards for railway signaling systems (SIL4). Important aspects that led us to choose the RBC are the suitable level of complexity (reasonably complex, but manageable) and that it essentially is a software system.

The RBC model has been developed using PTC Integrity Modeler and basically consists of SysML block definition diagrams and state charts as well as additional C++ program

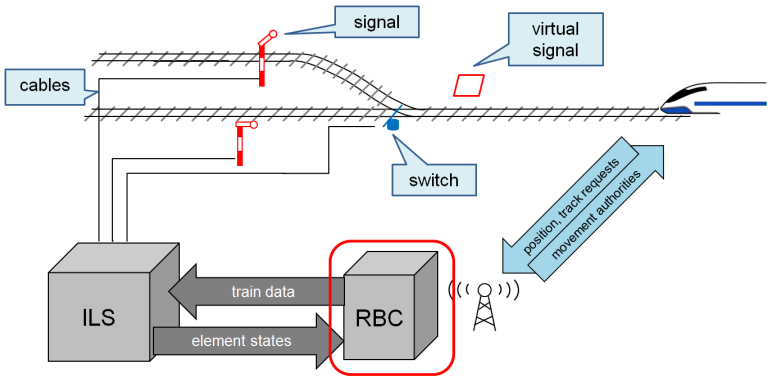

Fig. 1: The RBC as part of the ETCS Level 2 (simplified presentation, ILS = interlocking system).

code. The model exhibits no complete implementation of an RBC, only core functionality necessary to operate a demonstration line has been included. However, the functionality available has been modeled in a realistic scope; for example, the RBC is multi-train and multi-interlocking capable, can be used for arbitrary track layouts, and can be extended upon demand to include more special functionalities. For more information on the model as well as on the underlying standards, tools and languages we refer the reader to the articles [SH16; SHC17].

3 Approach

The railway simulation into which the model was to be integrated was given as the one available in DLR's RailSiTe® lab. It is regularly used for ETCS on-board unit tests, for which the lab is accredited, so it supports the simulation of train rides controlled by ETCS. Also, it is regularly used for studies conducted in a train driver simulator. This latter set-up was utilized for our simulation since it naturally supports validation of an RBC: The outcome of the most important RBC actions is visible on the driver machine interface (DMI), and the time of passing a balise group which triggers some RBC action can be detected from the front window.

Fig. 2 shows the working steps that have been performed in order to run the RBC model as part of DLR's railway simulation. The arrows indicate the order (dependencies) of the steps; iterations (backwards arrows) occur in many places in practice but have been omitted here to avoid a cluttered picture. First of all, the model to be integrated has been developed in a way that code could be automatically generated from it. On the lab side, the necessary interfaces have been prepared directly on program code level. All of those activities are generic (colored blue), i.e. they are independent of a particular railway infrastructure (track topology, signal locations, gradient and speed profiles etc.). This is important for signaling systems as a new development for each specific infrastructure would be unaffordable; but on

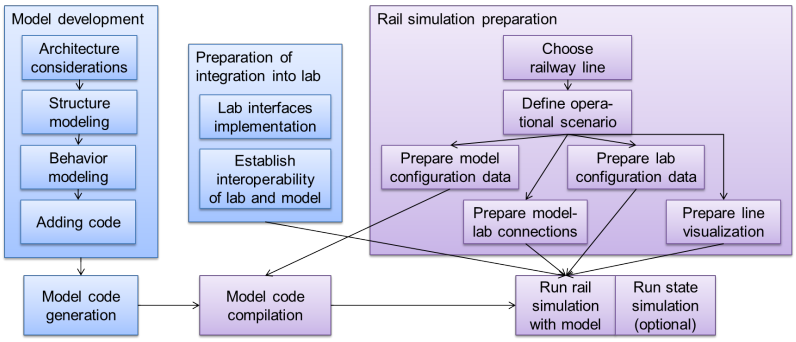

Fig. 2: Working steps towards running a signaling component model as part of a railway simulation. Generic development steps are colored blue, steps requiring a specific track layout are colored purple.

the other hand this poses challenges for system validation since the system can be configured for arbitrary infrastructures and needs to ensure safe operation in any case. For the purpose of this paper, we shall not discuss this issue further but limit ourselves to showing that we have kept our model generic and that it can indeed be configured for arbitrary infrastructures, as well as the lab (the corresponding activities are colored purple). Finally, the railway simulation of the lab can be run together with the model, i.e. a concrete operational scenario is simulated on the specific infrastructure for which the systems have been configured. As the modeling tool suite used provides state simulation capabilities, this can optionally be run in parallel, resulting in animated model diagrams during the railway scenario simulation.

4 Creating a Model Suitable for Integration into a Railway Simulation

As shown in Fig. 2, model development basically comprises four kinds of activities: architectural considerations, structure modeling, modeling of the behavior and adding program code to the model. The goal of integration into a railway simulation already influences the first activity, since it includes the definition of interfaces of the model, some of which may be connected to the simulation later. In the case of the RBC the latter applies to the interlocking and train (on-board unit) interfaces, cf. Fig. 3. Luckily, in both cases we could build on existing standards (the German SCI RBC [SCI14] for the interlocking interface and ETCS Subset 026 [SS16a] for the train interface) which precisely define the messages that can be exchanged. Those have been implemented in the model (and made available in the lab, see Sect. 6 below). While it made sense to choose the original messages, it was decided to deviate from the original underlying communication mechanisms: Instead of the DB RaSTA protocol via UPD/IP Ethernet communication demanded by the SCI RBC and instead of the GSM-R radio communication demanded by ETCS, for both interfaces the messages are transferred via TCP/IP Ethernet between model and lab. This choice did not only imply easier implementation of the communication, but also could be realized

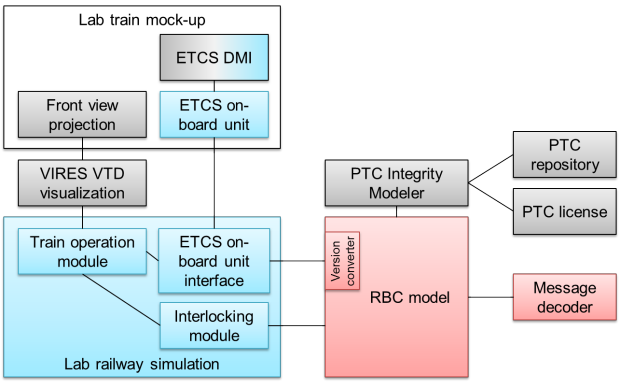

Fig. 3: Components and communication links for the integration of the RBC model into the railway simulation. The red components constitute the RBC, the light blue ones are the further relevant components of the railway simulation, and the grey ones pertain to visualization.

without special radio equipment. However, one needs to bear in mind the differences (e.g. concerning delay or reliability of transmission) that are introduced into the simulation compared to the original communication. In order to be able to work also with original components in the future, the RBC interfaces have been modeled in a flexible way so that arbitrary communication mechanisms may be plugged in.

One precondition on the model has been that it can be executed at least in real time, since this is desirable when including a train simulator in the railway simulation. This requirement needed to be kept in mind throughout all of the model development activities, as control and data flow are refined starting from interface definition, continuing with flat or deeply nested modeled structures and behavior which may use broadcasting or direct links, down to the use of more or less efficient data structures and algorithms on code level. Fortunately, real-time conditions for signaling systems are not that hard (usual time for a functionality in the overall signaling system is one to several seconds) and it turned out that so far there were no difficulties at all to meet the real-time requirement. However, this may become of interest again when the model is used for multi-train/multi-interlocking operation and further functionalities are included.

There are also some practical issues when running a railway component such as loading configuration data, getting diagnostic information and accessing the component for maintenance. Such issues are solved by manufacturers for real components operating in the real railway environment, but a different solution may be needed for models as part of a simulation. For easy configuration of the model, we decided to provide two separate files: One settings file containing identifier/value pairs for initialization during start-up, and one file containing infrastructure data simply in the form of program code which is included at compile time, creating several infrastructure objects. This way no additional data format needed to be defined and implemented. For diagnostic information, a flexible logging

mechanism has been included in the model. For interaction a simple command line menu has been prepared which provides a convenient way to establish or close RBC connections to individual components during runtime; for initial connections to be established automatically at start-up, default settings can be passed to the RBC via the settings file.

5 Code Generation

In principle, the code generation should be a push button activity and independent of the integration into the railway simulation, except that it is a first necessary step to transform the model into an executable form. In practice, it nevertheless is important to understand how the model is transformed to program code for various reasons:

First of all, code generation may only support part of the model diagrams and elements available in a language or modeling tool. The modeler needs to know this and to stick to that subset.

In case of a SysML model, there is no completely defined semantics available, but it is given to the model during code generation. For example, the execution order of state chart transitions may depend on the code generator. Thus understanding the code generator semantics is necessary if one wants to validate the model by running it in a railway simulation — it may be the case that some unexpected behavior originates from the code generator semantics rather than from a model error.

Another questions regards the handwritten code that has been added to the model: Is it integrated into the code generated from the model's diagrams as expected by the modeler? For example, under which conditions are initial and final parentheses generated for a code snippet, and is the handwritten code generated verbatim or is it processed in some way?

We have also seen cases where obviously the code generator produced erroneous code; sometimes the errors result in a compiler message, but sometimes they stay hidden and may be a subtle source of unexpected behavior. We utilized scripts in order to quickly, reliably and reproducibly remove such errors from the code files after each code generation. However, those scripts may require maintenance when the model is changed, so it is desirable to retrieve an updated code generator as soon as possible.

Finally, code generators may not only be capable of generating the mere product code, but also instrumented code for simulation of the model. We decided to use such a feature in order to have animated state charts available during the railway simulation. We did not use it for manipulation of the running model, which might have caused model behavior which is otherwise impossible to stimulate. Another issue is to make sure that the additional code in the instrumented version does not change the behavior of the model (except for intended manipulations by the user) in any way; in our case, the code generator documentation claimed this.

6 Preparation of Lab Interfaces

We have discussed the interfaces between model and lab from the model side already in Sect. 4. Of course, the lab needs to provide the counterpart of those interfaces. Since for a lab reuse of such interfaces is desirable, they should follow widespread standards, if available. As mentioned above, in our case for both relevant lab-RBC interfaces such standards exist; the ETCS standard had already been implemented in the lab and was then made available externally via TCP/IP, and the SCI RBC standard has been newly implemented for the RBC lab integration. The interface to the PTC Integrity Modeler was provided by PTC as part of the tool and of the instrumented code (RBC model interface side), based on TCP/IP communication as well.

For the RBC train interface further measures in order to ensure interoperability between lab and RBC have been necessary. This was due to an old version of the ETCS on-board unit which was incompatible with the RBC which was developed according to the newest version. For message decoding on the RBC side, an existing component (external to the model, see Fig. 3) was used which supports the different versions anyway. However, for further processing a version converter was needed. Although officially not foreseen in the ETCS standard, such a conversion was inferred from another ETCS specification ([SS16b], where data forwarding between RBC and a neighboring RBC of different versions is specified).

The use of the external message decoder component brought up the need for specification of that particular interface. It was decided to realize this on TCP/IP basis as well (which did not cause additional asynchronism problems since the messages passed to the decoder are received by the RBC from the on-board unit via TCP/IP right before). Messages were wrapped, adding IDs in order for the RBC to be able to distinguish which message coming back from the decoder has been received from which train previously.

7 Preparing a Railway Simulation

In order to run a railway simulation, a particular railway line (track topology and signaling equipment) and operational scenario (train type, train control system, route, course of events) have to be identified (cf. Fig. 2). In our case the single track line from Braunschweig main station to Braunschweig Gliesmarode station and the scenario of a train ride starting in ETCS Level 0 with later entry to Level 2 was chosen since for the latter a connection between train and RBC is set up and used. Since that particular line is not equipped with ETCS in reality, this has been done (mainly positioning of balises and assignment of balise messages) according to the applicable DB guidance document [Ril14].

The line and scenario data are then used to prepare the different parts of the simulation: Configuration files for the RBC model and the lab railway simulation are created which contain distances, signals, points, balises, speed and gradient profiles, etc. The 3D-world for visualization of the line has been available already but was adjusted to include the ETCS

balises. All of those preparations must happen in a consistent way; otherwise the intended scenario will not work properly e.g. due to triggering of automatic train stops or a time shift between simulation visualization and logic. The built-in configuration consistency check that was implemented in the RBC helps to detect faulty RBC configurations.

The connection set-up between the different components from Fig. 3 is dependent on the line and simulation scenario as well. For example, the number of interlockings connected to an RBC may vary. It is also dependent on the deployment of components in the lab; in our case, we needed to configure the IP addresses and ports for the TCP/IP connections between the computers involved, and ensure a suitable network and firewall configuration of all computers.

8 Running the Simulation

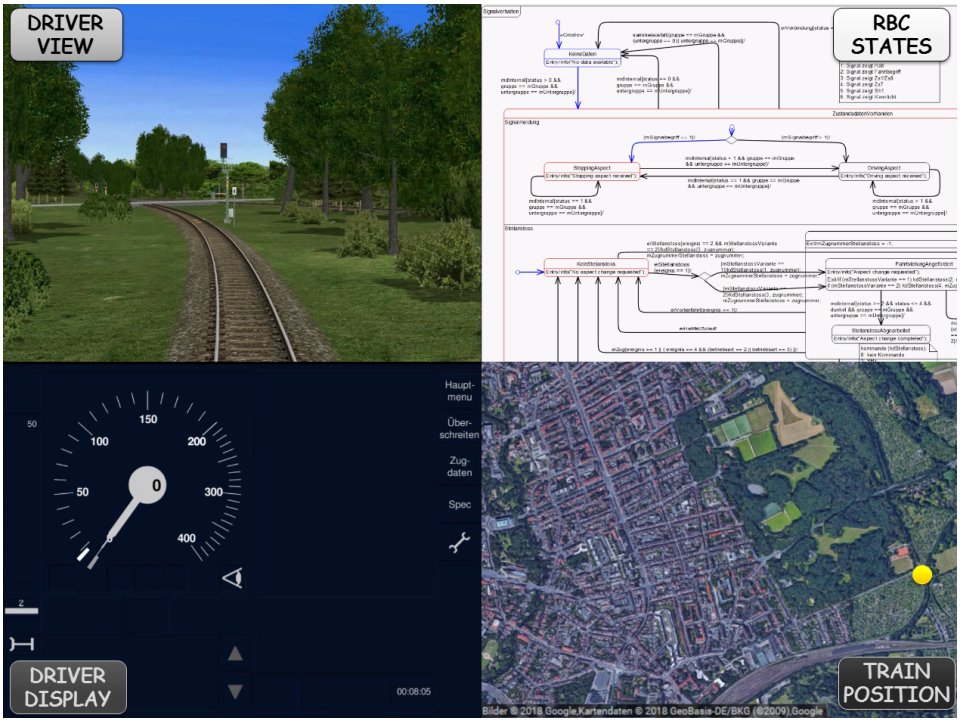

Fig. 4: Visual output of the railway simulation (driver's view and ETCS DMI on the left) and of the PTC Integrity Modeler state animation (upper right) as well as the current position of the train on the map (lower right). The state chart shows in red the current RBC internal state ("stopping aspect") of the red signal visible in the driver's view.

For running the RBC model as part of the railway simulation, at least all of the components from Fig. 3 that are not purely grey (pertain not to mere visualization) need to be started

up. In general, it is favorable that the component implementations do not create too many dependencies regarding the order of start-up and connect automatically to each other using predefined connection settings; for particular connections, exceptions from that rule may make sense. Afterwards, the components need to be brought to the correct initial state for the simulation scenario. In our case, that meant that a "Start of Mission"procedure bringing the ETCS on-board unit to Level 0 operation had to be executed via the ETCS DMI and that initial information on current signal and point states was transferred automatically from the interlocking module to the RBC after start-up of both components. For finally executing the scenario, in our set-up the train had to be driven manually (speed control lever in mock-up and interaction with ETCS DMI) and also routes had to be set manually (interaction with interlocking module). This offered an easy possibility to try out slight modifications of the original operational scenario. In this context, DMI and front view visualization become mandatory, of course. However, the lab could have been set-up to simulate the precise scenario automatically, if that had been desired.

The simulation can be run with or without the RBC state chart animation. In the first case, the instrumented version of the RBC model code must be generated, compiled and used. The PTC Integrity Modeler must be started (requiring the PTC license) and the RBC model must be loaded (requiring the PTC repository). Coherence must be ensured between the model versions loaded in Integrity Modeler and the one from which the running RBC model was generated. Then Integrity Modeler simulation mode can be started; starting up the RBC model as well will result in a connection between those two which is used to transfer information about newly created objects and changed states from the running model to Integrity Modeler. The latter visualizes that by opening and coloring state charts for those objects. Fig. 4 shows the visual outputs of the lab and the state animation for one scene of the running operational scenario.

9 Validation Results and Scope

Here we shortly describe our experiences with the RBC model validation by running it as part of the RailSiTe® railway simulation. We used the visual lab output as well as the state animation and the RBC logging function in order to check whether the scenario proceeded as expected. This way we could detect errors from a wide range of sources: errors in the model, in the added code, in the code generator, in the newly implemented lab interfaces and in the configuration data. Most often it was not difficult to locate the component where the problem originated from; in order to support this, we ran the generated RBC code in a debugging tool. And we could do so step by step, each time advancing a bit further in the simulation scenario. Thus we conclude that validation by simulation is an effective tool which can provide evidence that the generic model as well as the configuration data used for the integration are valid. During validation it pays off if restarting the railway simulation and the model is easily possible.

The clear majority of errors we found was in hand-coded parts (e.g. the code added to the model). Since a good portion of the generated RBC code was generated from the modeled diagrams (estimation: nearly half of the code lines), this confirms that modeling and using code generation increases software quality. We found only very few errors in the modeled diagrams, e.g. a missing state chart transition; here the modeling proved to be advantageous again since it eased the location and correction of such errors. Problems that involved the code generation (different interpretation of model by user and code generator) were comparatively difficult to locate, but occurred only seldom.

10 Reflection on the Model in Simulation Approach

As stated in the paper's abstract, a railway simulation is one possible environment for model validation; other alternatives may include using stubs or running the model in a real environment. Tab. 1 provides a compact comparison. While working with stubs may

Criterion	Stubs	Simulation	Real Environment
Availability	high — automatic generation possible	low — special purpose software	medium — real hard- and software
Integration effort	low — mostly automatic	medium to high — integration effort, depending on available interfaces	medium — installation effort
Testing effort	high — in-/outputs on detailed level	low to medium — in-/outputs on high level	low — stimulation/read-off at external interfaces
Flexibility	high — direct manipulation	medium to high — indirect manipulation on different levels	low — manipulation only at external interfaces
Suited for complex scenarios	low — inputs too detailed	high — supports manipulation on high level	high — supports manipulation on high level
Closeness to reality	low — depends on user	high — realistic components	high — real components

Tab. 1: Advantages and disadvantages of different validation environments for railway signaling component models.

be a good idea for unit and integration testing, they do not provide (realistically) the environment's logic. On the other side, a real environment does, but may not be available or too costly, especially when it comes to a railway signaling environment. So all in all we think that for the validation of signaling component models railway simulations are a favorable choice.

11 Conclusion and Future Work

In this article, we have reported on the validation of a railway signaling component model by integrating it into a railway simulation. More precisely, a SysML model of an RBC has been integrated and validated into DLR's RailSiTe® laboratory.

A first conclusion, from the mere fact that the model could be integrated and ran well in the simulation environment, is the general feasibility of modeling railway signaling components generically and for realistic applications. Further steps to support this statement would be to try the same approach with different components, with different configurations, and with complete component models. Some work remains to be done if the development approach is to be applied for real signaling components: A detailed semantics for SysML (which may be part of the upcoming SysML 2.0 standardization work) would be needed as well as a concept for migration to the new approach and a certified code generator. Most probable, this will imply further restrictions on the model artifacts and programming language constructs used. At least the model at hand was developed to result in deterministic code (i.e. without concurrency) and avoiding circular control flow between the state charts.

Depending on the interfaces available and the configuration of the lab for a particular operational scenario, the integration of a model into a railway simulation may require some effort; certainly, there is potential for more efficient or automated ways to distribute configuration data from a single source to the different components of the simulation. However, we were pleased to see that the combination of model and railway simulation allowed for efficient validation: The model obviously resulted in a high initial quality of the component and provided the possibility for visual inspection during simulation by using the state animation feature; and the simulation allowed for convenient stimulation and output retrieval on the high-level interfaces like DMI and line visualization, while offering the possibility to dig deeper when erroneous behavior was observed. It also turned out that the inherent parallel validation of the configuration data and the newly implemented lab interfaces was not much of a problem since the assignment of errors to the different possible sources was quickly found in most cases, whereas tracking errors related to code generation was more difficult. All in all, we conclude that validation of models by running them in railway simulations is a practically feasible and may even be the favorable approach for overall component validation. Provided that some particular requirements of the approach, which we aimed to describe in this paper, are taken into consideration early, lab integration can be organized in a smooth way.

In the future, we would like to investigate and exploit the benefits of a validated model. It may be used as a reference for real signaling components, or for test case generation for such components. It would be nice to see whether the quality of the component can be increased further e.g. due to the extra model coverage criteria generated test cases can fulfill. We also expect a clear decrease in maintenance effort of test cases (simply change model and generate again).

References

[EUL18] EULYNX Website, 2018, URL: https://www.eulynx.eu, visited on: 02/19/2018.

[Ril14] LST-Anlagen planen: Grundsätze zur Erstellung der Ausführungsplanung PT1 für ETCS Level 2, tech. rep. Richtlinie 819.1344, draft, DB Netze AG, Apr. 24, 2014.

[SCI14] SCI-RBC – FAS TAS TAV Schnittstelle ESTW-RBC V2. Baseline 0.19, tech. rep., DB Netze AG, June 6, 2014.

[SH16] Schwencke, D.; Hungar, H.: Development of Reference Models of Signaling Components: the Example of the Radio Block Center. Signal+Draht 108/9, pp. 24–32, 2016.

[SHC17] Schwencke, D.; Hungar, H.; Caspar, M.: Between Academics and Practice: Model-based Development of Generic Safety-Critical Systems. In (Huhn, M.; Hungar, H.; Riebisch, M.; Voss, S., eds.): Modellbasierte Entwicklung eingebetteter Systeme XIII (Proceedings of the Dagstuhl MBEES 2017 workshop). fortiss GmbH, Munich, pp. 1–18, 2017, URL: http://download.fortiss.org/public/mbees/mbees2017_proceedings.pdf.

[SS16a] SUBSET-026 – System Requirements Specification. Version 3.6.0, tech. rep., ERTMS, June 15, 2016, URL: http://www.era.europa.eu/Core-Activities/ERTMS/Pages/Set-of-specifications-3.aspx.

[SS16b] SUBSET-039 - FIS for the RBC/RBC Handover. Version 3.2.0, tech. rep., ERTMS, June 15, 2016, URL: http://www.era.europa.eu/Core-Activities/ERTMS/Pages/Set-of-specifications-3.aspx.