Entwicklung von Referenzmodellen von LST-Komponenten am Beispiel der Streckenzentrale

Development of reference models of signalling components: the example of the radio block centre

Daniel Schwencke | Hardi Hungar

Die Verwendung von Modellen ermöglicht es, bereits in frühen Phasen des Entwurfs, das Verhalten eines Systems sehr genau zu beschreiben. Solche Modelle können als Referenz sowohl späteren Entwurfsschritten als auch der finalen Implementierung dienen. Im vorliegenden Artikel wird über die Modellierung von Anteilen der ETCS-Streckenzentrale bzw. ihrer Schnittstelle zum Stellwerk berichtet. Aus dem entstandenen Modell kann automatisch ein Code generiert werden, der unter anderem für die Einbindung in eine Bahnsimulation geeignet ist. Die Modellierungsmethode und die Erfahrungen bei der Modellerstellung werden vorgestellt.

1 Einleitung

Bereits in frühen Phasen der Entwicklung eines Systems können Modelle zur Beschreibung des Systemverhaltens verwendet werden. Der formale oder zumindest semiformale Charakter der Modelle erlaubt es, eine hohe Präzision in der Formulierung zu erreichen. Und er bietet einen Ausgangspunkt für eine Anzahl von Verfahren, mit denen Entwurfsaktivitäten unterstützt oder sogar automatisiert werden können. Insbesondere sind dies Validierung, Verifikation und Code-Erzeugung.

In diesem Artikel werden Aspekte der Modellierung einer Streckenzentrale (Radio Block Center, RBC) vorgestellt. Ein RBC stellt im European Train Control System (ETCS) die Funkschnittstelle zwischen Zug und Stellwerk dar (Bild 1). Die Aktivitäten sind auf die Erstellung eines Systems ausgerichtet, das als Referenz eines RBC im ETCS dienen kann. Zum einen soll das Modell als Spezifikation in Lastenheften verwendet werden können und dabei

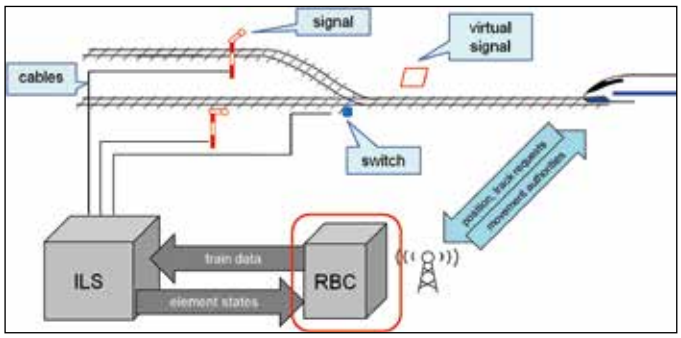


Bild 1: Das RBC als Komponente des ETCS Level 2/3 (vereinfachte Darstellung, ILS = Stellwerk)
Fig. 1: The RBC as part of the ETCS level 2/3 (simplified presentation, ILS = interlocking system)

The use of models enables a very precise description of system behaviour already in early design stages. Such models may serve as a reference for further design steps as well as for the final implementation. This article reports on the modelling of parts of the ETCS radio block centre or, more precisely, of its interlocking interface. From the model developed, it is possible to generate code automatically which is, amongst other uses, suitable for integration in a railway simulation. The modelling method and experience from the development of the model are presented.

1 Introduction

Already in early stages of system development, models can be used for capturing system behaviour. The formal or at least semi-formal nature of models makes it possible to reach a high precision in the formulation. Additionally, it constitutes the basis for a number of techniques which can support, or even automate, design activities. These include, in particular, validation, verification and code generation.

This article presents aspects of modelling a radio block centre (RBC). Being part of the European Train Control System (ETCS), an RBC constitutes the radio interface between train and interlocking system (fig. 1). The work aims at creating a system that can serve as a reference of an RBC in the ETCS. On the one hand, the model is supposed to be utilizable in product specifications and substantially support their completeness and correctness, as demanded, for instance, by the new German type approval ("Neue Typzulassung" or NTZ, [1]). On the other hand, after annotation with program instructions and addition of further components, program code can be generated from the model. The code can be configured for a railway line, becoming an implementation. The implementation in turn can be validated in simulations and then provide a behavioural reference. For the time being, only part of the RBC functionality has been modelled. This article presents the current state of the work.

The model development is related to activities of the German railway infrastructure manager, DB Netz AG, and further European infrastructure managers towards the standardisation of interfaces in the interlocking area (the DB project "Neuausrichtung der Produktionssteuerung", NeuPro for short, and the "European Initiative Linking Interlocking Subsystems", EULYNX for short, see [2] for the latter). Results from those activities have already successfully served as patterns for parts of the model.

wesentlich dazu beitragen, die Vollständigkeit und Richtigkeit zu erreichen, wie sie beispielsweise im Prozess der Neuen Typzulassung (NTZ) gefordert werden [1]. Zum anderen kann aus dem Modell nach seiner Annotation mit Programminstruktionen und Ergänzung durch zusätzliche Komponenten ein Programmcode erzeugt werden. Der Code kann auf Strecken projektiert werden und so zu einer Implementierung werden. Die Implementierung kann in Simulationen validiert werden und steht dann als Verhaltensreferenz zur Verfügung. Zum jetzigen Zeitpunkt sind nur Teile der RBC-Funktionalität so modelliert worden. Dieser Artikel stellt den aktuellen Stand dar.

Die Modellierung steht im Zusammenhang mit Aktivitäten der DB Netz AG und weiterer europäischer Eisenbahninfrastrukturbetreiber zur Vereinheitlichung der Schnittstellen im Stellwerksbereich („Neuausrichtung der Produktionssteuerung", kurz NeuPro, bzw. „European Initiative Linking Interlocking Subsystems", kurz EULYNX [2]). Aus diesem Umfeld stammen bereits Vorlagen für Teile des Modells.

Im Folgenden wird in Abschnitt 2 zunächst ein Überblick über die Grundlagen der Modellentwicklung gegeben (relevante Spezifikationen, Sprachen und Werkzeuge). Abschnitt 3 beschreibt das entstandene Modell mit seiner Architektur und seinem Verhalten. Bisherige Tests des Modells werden in Abschnitt 4 behandelt. In Abschnitt 5 werden weitere Schritte diskutiert und in einem Fazit der gewählte modellbasierte Ansatz reflektiert.

2 Rahmen der Modellierungsaktivitäten

2.1 Relevante Spezifikationen

Das ETCS wird durch die Eisenbahnagentur der EU (früher: Europäische Eisenbahnagentur), die Union Industry of Signalling (UNISIG) sowie die European Rail Traffic Management System (ERTMS) Users Group in einer Vielzahl von Dokumenten, den sogenannten Subsets, spezifiziert. Zentral ist die Systemanforderungsspezifikation [3]; daneben existieren diverse Spezifikationen für einzelne Systemkomponenten und Schnittstellen. Für das RBC ist die funktionale Schnittstellenspezifikation für die Übergabe von Zügen zwischen RBC [4] von Bedeutung. Eine Spezifikation des RBC als solches ist jedoch nicht Teil der Subsets, sondern den jeweiligen Herstellern – unter Berücksichtigung der europäischen Vorgaben – überlassen.

ETCS wird in verschiedenen Staaten in unterschiedlichen Umfängen und Parametrisierungen verwendet. In Deutschland ist dies im Lastenheft [5] der DB Netz niedergelegt. Schließlich wurde im Rahmen der NeuPro-Aktivitäten bei DB Netz die Schnittstelle zwischen elektronischem Stellwerk (ESTW) und RBC, SCI-RBC genannt, spezifiziert [6]. Solche nationalen Vorgaben sind für den Hersteller eines RBC für das jeweilige Land von Bedeutung.

Die Abstraktionslevel der Spezifikationen sind dabei sehr unterschiedlich: selbst innerhalb der europäischen Spezifikationen reicht das Spektrum von funktionalen Anforderungen über Systemzustände und Abläufe bis hin zu konkreten Datenformaten und -werten, je nachdem, ob ein Systembestandteil für die europäische Interoperabilität von Bedeutung ist. Die Erstellung eines Modells ist umso einfacher, je ausdifferenzierter (und logisch strukturierter) die zugrundeliegende Spezifikation bereits ist. Während die meisten der genannten Spezifikationen vor allem textuelle Anforderungen enthalten, hat die SCI-RBC bereits einen semiformalen Charakter. Neben erklärendem Text wurde hier das verbindliche Verhalten der Schnittstelle in Zustandsdiagrammen spezifiziert.

In the remainder of this article, first an overview of the general framework of the model development (relevant specifications, languages and tools) is given in section 2. In section 3, the resulting model is described including its architecture and behaviour. The testing of the model to date is dealt with in section 4. Section 5 concludes with a presentation of future plans and a preliminary evaluation of the model-based approach.

2 General framework of the modelling activities

2.1 Relevant specifications

The ETCS is specified by the EU Agency for Railways (formerly European Railway Agency), the Union Industry of Signalling (UNISIG) as well as the European Rail Traffic Management System (ERTMS) Users Group in a variety of documents, the so-called subsets. The central document is the system requirements specification [3], which is complemented by various specifications for single system components and interfaces. For the RBC, the functional interface specification for the handover of trains between RBC [4] is of importance. A specification of the RBC as such, however, is not part of the subsets but left to the individual manufacturers, for which they have to take into account the European specifications.

ETCS is used in different countries to different extents and with different parametrisations. In Germany these are laid down in the specification [5] of DB Netz. Finally, as part of the NeuPro activities, DB Netz specified the interface between an electronic interlocking system (ILS) and an RBC, called SCI-RBC [6]. Such national specifications are of importance for the manufacturer of an RBC for the country concerned.

The specifications differ considerably in their level of abstraction. Even the European specifications range from functional requirements over system states and procedures to concrete data formats and values, depending on the importance for European interoperability of a system component. The creation of a model is simpler, the more differentiated (and logically structured) the underlying specification. Whereas most of the aforementioned specifications mainly consist of textual requirements, the SCI-RBC is already of semi-formal nature. Besides explanatory text, the mandatory behaviour of the interface is specified in state diagrams here.

2.2 Approach

The DB specification SCI-RBC largely uses SysML (Systems Modelling Language) diagrams and thus represents a good starting point for modelling the RBC. Besides other types of diagrams, it is mainly state machines that are used. They bindingly specify the behaviour defined by the semantics of the diagram elements, but not the use of such diagrams or their architecture in the further development. The system actions in the state machines however are only given in textual form and are not yet formalised precisely. Consequently, it is not possible to generate code directly from those state machines. Furthermore some requirements which cannot be expressed in diagrams remain in textual form.

For the formal modelling of the RBC it seemed sensible to use SysML as well. Besides being able to stay close to the specification, SysML has the advantage of being widespread and having rich support by modelling tools, code and test case generators. As target language for code generation from the model an object oriented language is favourable, enabling the configura-

2.2 Vorgehen

Die DB-Spezifikation SCI-RBC verwendet weitgehend Diagramme der SysML (Systems Modeling Language) und ist somit ein guter Ausgangspunkt für die Modellierung des RBC. Neben weiteren Diagrammarten werden hauptsächlich Zustandsmaschinen genutzt. Verbindlich spezifiziert wird dabei das durch die Bedeutung (Semantik) der Diagrammelemente festgelegte Verhalten, nicht jedoch die Verwendung solcher Diagramme oder ihrer Architektur in der weiteren Entwicklung. Die Aktionen des Systems sind in den Zustandsmaschinen allerdings nur textuell bezeichnet und noch nicht präzise formalisiert. So lässt sich aus diesen Zustandsmaschinen kein Code generieren. Daneben verbleiben auch einige nicht in den Diagrammen abbildbare Anforderungen in Textform.

Für die formale Modellierung des RBC bot es sich an, ebenfalls SysML zu verwenden: neben der Nähe zur Spezifikation spricht dafür, dass die Sprache weit verbreitet ist und es dafür Modellierungstools, Code- und Testfallgeneratoren gibt. Als Zielsprache für die Codegenerierung aus dem Modell empfahl es sich, eine objektorientierte Sprache zu verwenden, um die Projektierung eines (generischen) RBC für verschiedene Bahninfrastrukturen durch die Instanziierung von Klassen abbilden zu können; hier wurde C++ gewählt. Als passendes Tool für Modellierung und Codegenerierung wurde der PTC Integrity Modeler (bis 2014: Atego Modeler) zusammen mit dem PTC Automatic Code Synchronizer (vormals Artisan Studio Automatic Code Synchronizer) identifiziert. Der Code Synchronizer unterstützt dabei die Codegenerierung für SysML-Blockdefinitionsdiagramme/Interne Blockdiagramme und SysML-Zustandsdiagramme.

Um die Schnittstelle auch unabhängig von einem ESTW betreiben zu können, wurde entschieden, neben dem für das RBC benötigten Schnittstellenende auch das ESTW-seitige Schnittstellenende mitzumodellieren.

3 Modellierung der Schnittstelle zum Stellwerk

3.1 Modellarchitektur

Wie in Bild 2 dargestellt, ist die Schnittstelle zwischen ESTW und RBC in das ESTW-seitige und RBC-seitige Schnittstellenende sowie in das Übermittlungsmedium dazwischen gegliedert. Die Schnittstellenenden sind Teil des RBC bzw. der ESTW-Zentraleinheit.

Jedes Schnittstellenende muss die Möglichkeit bereitstellen, ein logisches Abbild der für die Schnittstelle relevanten Bahninfrastruktur aufzubauen (die Schnittstelle zu projektieren), die Verbindung zum jeweils anderen Schnittstellenendpunkt zu verwalten, von dort eingehende Telegramme zu verarbeiten, nach dort ausgehende Telegramme zu versenden sowie Ereignisse aus dem ESTW- bzw. RBC-Kern zu verarbeiten und dorthin Meldungen weiterzuleiten. Dieser Funktionsumfang wurde in je einem Hauptpaket des Schnittstellenendes zusammengefasst (obere Pakete in Bild 3). Durch die klare Trennung der Schnittstellenenden ist es

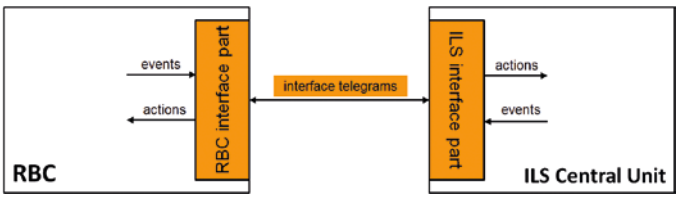


Bild 2: Die modellierte Schnittstelle zwischen ESTW und RBC (farbig hinterlegt)
Fig. 2: The modeled interface (colored parts) between interlocking and RBC

tion of a (generic) RBC for different railway infrastructures by means of instantiation of classes; here C++ was chosen. Regarding a suitable tool for modelling and code generation, PTC Integrity Modeler (through 2014: Atego Modeler) together with PTC Automatic Code Synchronizer (formerly Artisan Studio Automatic Code Synchronizer) was identified. The Code Synchronizer supports code generation for SysML block definition diagrams/internal block diagrams and SysML state machine diagrams.

In order to be also able to run the interface independently of an ILS it was decided to model the ILS interface part as well as the required RBC interface part.

3 Modelling the interlocking interface

3.1 Model architecture

As shown in fig. 2, the interface between ILS and RBC is divided into the ILS and RBC interface parts and the information transmission in between. The interface parts belong to the ILS and RBC respectively.

Each interface part needs to provide the functionality to build up a logical image of the railway infrastructure relevant for the interface (i. e. to configure the interface), to manage the connection to the opposite interface part, to process incoming telegrams from that part, to send outgoing telegrams to that part and also to process events from the ILS or RBC core and to forward messages to there. This functional range was gathered in one main package for each interface part (see the top packages in fig. 3). Thanks to the clear separation of the interface parts, it is easily possible to build either the complete interface or the separate interface parts from the code generated for the packages. These packages comprise the SysML share of the model. In addition to them, there are further packages whose contents mainly have program character.

One commonly used package provides basic data types for the model, another one contains the functionality for parsing the data from the telegrams transmitted via the interface, for checking that data and for composing data into telegrams. Other packages provide features (not specified by SCI-RBC) that are useful for testing and running the interface (for the simulation of events external to the interface and for the logging of interface states).

The block structure of the main package is similar for both interface parts; it is the RBC part (fig. 4) that is considered below. The “ETCS-Z” block represents the central access point. It abstracts away from the transmission channel of the interface, i. e. the operation for sending telegrams can be flexibly implemented in a derived block. As a default, the derived “Referenz-ETCS-Z” block is provided which only logs outgoing telegrams. Additionally, a variant was prepared which transmits the telegrams through a TCP/IP connection. In that way, the RBC can be run in different environments.

The RBC manages an arbitrary number of field elements, i. e. points, signals and train data notification spots. Properties shared by all field elements are modelled in the “Feldelement-ETCS-Z” block, individual properties in the derived blocks “Weichenelement-ETCS-Z”, “Signalelement-ETCS-Z” and “ZDMPElement-ETCS-Z”. This construction makes it possible to forward events or telegram information to single field elements as well as (in a uniform way) to certain groups or all field elements. The other way round, single field elements can, for

thalesgroup.com

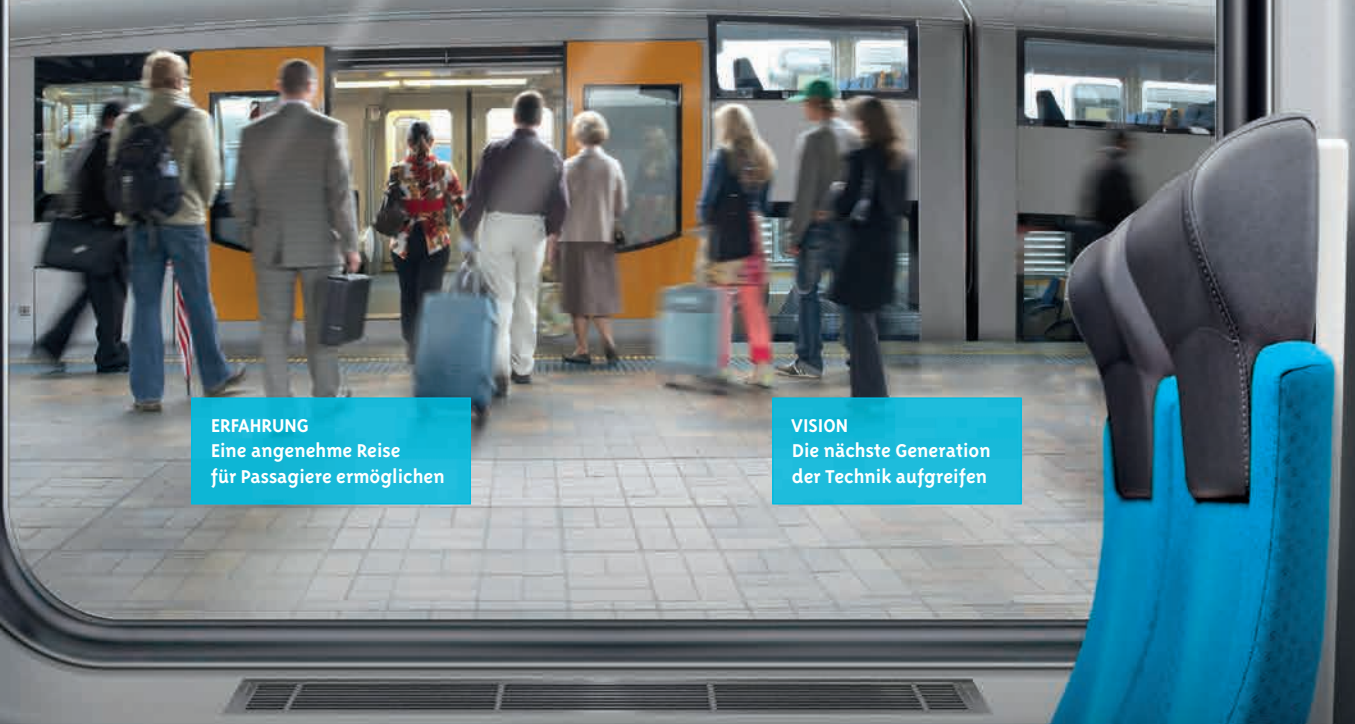
Transportlösungen

Wann immer es auf Sicherheit ankommt, haben wir die richtige Antwort

LEISTUNG
Steigerung der Verfügbarkeit, Zuverlässigkeit und Kapazität

KONNEKTIVITÄT
Garantierte nahtlose Mobilität für Menschen und Waren

SICHERHEIT
Schutz Ihrer kritischen Infrastruktur



ERFAHRUNG
Eine angenehme Reise für Passagiere ermöglichen

VISION
Die nächste Generation der Technik aufgreifen

Im Bereich Transport werden jeden Tag Millionen von kritischen Entscheidungen getroffen. Mit der Fähigkeit, komplexe technische Projekte effizient zu leiten, spielt Thales dabei eine zentrale Rolle. Wir bieten Signaltechniklösungen für Nah- und Fernverkehr, Kommunikations- und Überwachungstechnologie, Systeme für den Fahrgeldeinzug sowie Support für die Wartung. Durch die Expertise von Thales können Sie Cyberbedrohungen rechtzeitig begegnen. Wann immer es auf Sicherheit ankommt, hat Thales die richtige Antwort.

Suche: Thalesgroup

THALES
Together • Safer • Everywhere

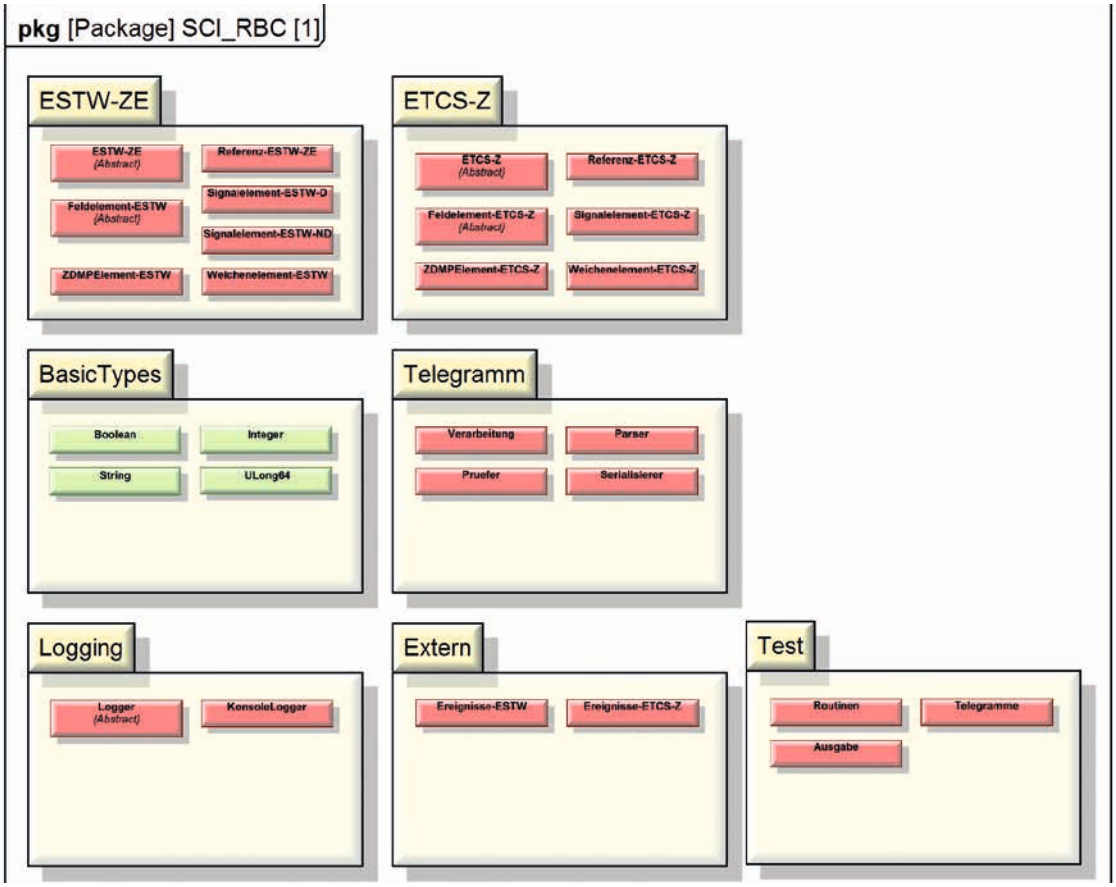


Bild 3: Paketstruktur des Modells im SysML-Paketdiagramm
Fig. 3: Package structure of the model as SysML package diagram

einfach möglich, aus dem daraus generierten Code wahlweise die Gesamtschnittstelle oder die einzelnen Schnittstellenenden zu erstellen. Diese Pakete enthalten die SysML-Anteile der Modellierung. Daneben gibt es weitere Pakete, deren Inhalte vorwiegend die Form von Programmen haben.

Ein gemeinsames Paket stellt grundlegende Datentypen für das Modell bereit; ein anderes die Funktionalität zur Extraktion der Daten aus den über die Schnittstelle übertragenen Telegrammen, zu ihrer Überprüfung und ihrer Zusammensetzung zu Telegrammen. Weitere Pakete stellen für Test und Betrieb der Schnittstelle nützliche (nicht SCI-RBC-spezifizierte) Features zur Simulation schnittstellenexterner Ereignisse und zum Loggen von Schnittstellenzuständen zur Verfügung.

Die Blockstruktur der Hauptpakete der jeweiligen Schnittstellenendpunkte ist ähnlich; im Folgenden wird die RBC-Seite (Bild 4) betrachtet: der Block „ETCS-Z" stellt den zentralen Zugriffspunkt dar. Er abstrahiert vom Übertragungskanal der Schnittstelle, d. h. die Operation zum Senden von Telegrammen kann in einem abgeleiteten Block flexibel implementiert werden. Standardmäßig wird der abgeleitete Block „Referenz-ETCS-Z" bereitgestellt, der zu versendende Telegramme lediglich loggt; es wurde jedoch auch eine Variante erstellt, die die Telegramme über eine TCP/IP-Verbindung verschickt. Somit kann das RBC in den verschiedensten Umgebungen betrieben werden.

Das RBC verwaltet eine beliebige Zahl an Feldelementen, d. h. Weichen, Signale und Zugdatenmeldepunkte. Gemeinsame Eigenschaften sind im Block „Feldelement-ETCS-Z" modelliert, individuelle Eigenschaften in den abgeleiteten Blöcken „Weichenelement-ETCS-Z", „Signalelement-ETCS-Z" und „ZDMPElement-ETCS-Z". Diese Konstruktion ermöglicht es, Ereignisse oder Telegramminformationen sowohl an einzelne Feldelemente als auch

example, pass data which is to be used for an outgoing telegram to the "ETCS-Z" block.

In conclusion, the architecture presented above respects principles like encapsulation, reusability and uniformity, which make for a clear model structure and good maintainability. At the same time, the model is flexible through admitting variants in certain places. Finally, the performance aspect has been taken into account in design decisions.

3.2 Modelling behaviour

The typical course of action on the interface between ILS and RBC starts with an external event which causes one interface part to change its state. Subsequently, a telegram is transmitted to the other interface part, which in turn changes its state. For example, the event "point W1 locked in left position" is notified to the ILS. It is forwarded to the logical representation of point W1 within the ILS interface part, which changes to the state "locked in left position". This in turn triggers the transmission of a point status telegram. When this is received by the RBC interface part, the telegram information is forwarded within that part to the logical representation of point W1, which also changes state to "locked in left position".

Large parts of the interface behaviour can be represented in state diagrams associated with the respective blocks, such as the sequence of the initial loading of the RBC by the ILS in the state diagram of the "ETCS-Z" block or the notified position of a point in the state diagram of the "Weichenelement-ETCS-Z" block (fig. 5). For other parts, such as parsing the telegrams, it makes more sense to implement them directly in the target language of the code generation, in this case C++. Altogether, it can be stated that clearly more than half of the

in einheitlicher Weise an bestimmte Gruppen oder alle Feldelemente weiterzuleiten. Andersherum können einzelne Feldelemente z. B. zu versendende Telegrammdaten an den Block „ETCS-Z" geben.
Insgesamt berücksichtigt die Architektur damit Prinzipien wie Kapselung, Wiederverwendung und Einheitlichkeit, die für eine klare Modellstruktur und eine gute Wartbarkeit sorgen. Zugleich ist das Modell flexibel, indem es an definierten Stellen Varianten zulässt. Bei Designentscheidungen wurde einbezogen, dass das RBC eine gewisse Performanz aufweisen muss.

3.2 Verhaltensmodellierung

Ein typischer Ablauf auf der Schnittstelle zwischen ESTW und RBC besteht aus einem externen Ereignis, das an einem Schnittstellenende einen Zustandswechsel hervorruft. In Folge wird ein Telegramm an das andere Schnittstellenende übertragen, das seinerseits den Zustand wechselt. Beispielsweise wird dem ESTW das Weichenereignis „Weiche W1 in Linkslage verschlossen" gemeldet. Dieses Ereignis wird im ESTW-seitigen Schnittstellenende an die logische Repräsentation der Weiche W1 weitergeleitet, die in den Zustand „links überwacht" wechselt. Diese stößt wiederum den Versand eines Weichenmeldungstelegramms an. Wird dies vom RBC-seitigen Schnittstellenende empfangen, so wird die Telegramminformation an die dortige logische Repräsentation der Weiche W1 weitergeleitet, die ebenfalls in den Zustand „links überwacht" wechselt.
Große Teile des Verhaltens der Schnittstelle lassen sich entsprechend in zu den jeweiligen Blöcken gehörigen Zustandsdiagrammen abbilden: z. B. der Ablauf der anfänglichen Aufrüstung des RBC durch das ESTW im Zustandsdiagramm des Blocks „ETCS-Z" oder die gemeldete Weichenlage im Zustandsdiagramm des Blocks „Weichenelement-ETCS-Z" (Bild 5). Für andere Teile wie die Datenextraktion aus den Telegrammen ist es sinnvoller, sie direkt in der Zielsprache der Codegenerierung, hier C++, zu implementieren. Insgesamt lässt sich festhalten, dass deutlich über die Hälfte des Programmcodes der Schnittstelle aus SysML-Modellelementen generiert ist.

program code of the interface is generated from SysML elements.
By translating the state diagrams to executable programs, any code generation implements a specific semantics of the diagrams. The Automatic Code Synchronizer uses a so-called "run to completion" semantics, which executes, without interruption, a transition from the state diagram together with its non-event-triggered successor transitions and all actions specified within. This implies deterministic behaviour of the model, and requires the modeller to take specific care in order to avoid deadlocks. To achieve a high performance of the RBC code and traceability of the behaviour, no asynchronous elements have been used in the model. The only exception to that principle has been made in the parts dealing with incoming telegrams and events. These may be processed in an order different from that of their receipt, because they are cyclically retrieved from a telegram buffer and event queue respectively.

4 Model verification and validation

A model that is going to serve as reference has to be especially well tested. All of the tests described below were conducted through the execution of the program obtained after code generation and compilation. To this end, different calls of test routines were added to the generated code in the main function before compilation.
In a first step, telegrams and routines have been added to a separate package of the model (the right bottom package in fig. 3) for testing telegram parsing, telegram data checking and composition of telegrams.
As a second step, further telegrams and routines have been added in order to test the telegram processing of the single interface parts. This includes positive testing for the most common telegram messages as well as tests of erroneous situations, where the selection of the latter was done along the lines of the SCI-RBC test specification [7]. For that second step, both interface parts have been configured according to a fictitious station and

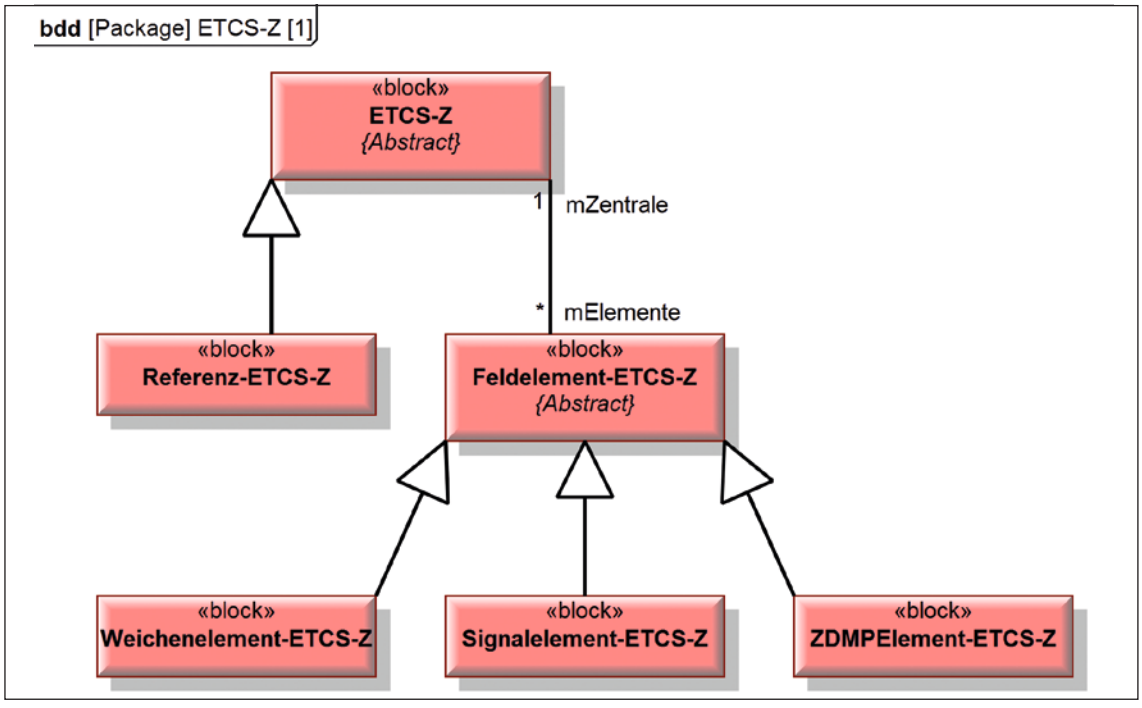


Bild 4: Blockstruktur des RBC-seitigen Schnittstellenendes im SysML-Blockdefinitionsdiagramm
Fig. 4: Block structure of the RBC interface part as SysML block definition diagram

Da die Codegenerierung die Zustandsdiagramme „operationalisiert", legt sie zwangsweise eine bestimmte Bedeutung (Semantik) der Diagramme fest. Der Automatic-Code-Synchronizer legt eine sogenannte „run to completion"-Semantik zugrunde, die eine Transition im Zustandsdiagramm inklusive möglicher folgender nicht ereignisgesteuerter Folgetransitionen und aller dabei spezifizierten Aktionen als Block abarbeitet. Das sorgt für ein deterministisches Verhalten des Modells, erlegt dem Modellierer jedoch Beschränkungen auf, um Deadlocks zu verhindern. Mit Blick auf die Performanz des RBC und die Nachvollziehbarkeit des Verhaltens wurden keine asynchronen Elemente bei der Modellierung verwendet. Einzig die in dem Schnittstellenendpunkt einkommenden Telegramme und Ereignisse werden potenziell nicht in der Reihenfolge ihres Eingangs abgearbeitet, da sie getrennt aus einem Telegrammpuffer bzw. einer Ereigniswarteschlange zyklisch abgefragt werden.

4 Modellverifikation und -validierung

Gerade ein Modell, das später als Referenz dienen soll, muss selbst gut getestet sein. Sämtliche im Folgenden beschriebenen Tests fanden durch Ausführen des nach Codegenerierung und Kompilierung erhaltenen Programms statt. Dazu wurden dem generierten Code vor der Kompilierung in der Hauptmethode entsprechende Aufrufe von Testroutinen hinzugefügt.
In einer ersten Stufe wurden dem Modell in einem separaten Paket (Bild 3 rechts unten) Testtelegramme und -routinen hinzugefügt, mit denen die Datenextraktion aus den Telegrammen, die Datenprüfung und das Zusammensetzen von Daten zu Telegrammen geprüft werden konnte.
Als zweites wurden weitere Testtelegramme und -routinen ergänzt, um die Telegrammverarbeitung in einem einzelnen Schnittstellenende zu prüfen. Dies beinhaltete Positivtests für die gängigsten Telegrammnachrichten sowie Tests von Fehlersituationen, wobei man sich bei der Auswahl letzterer an der Testspezifikation zur SCI-RBC [7] orientierte. Für diesen zweiten Schritt wurden die Schnittstellenenden gemäß eines fiktiven Bahnhofes projektiert und das erwartete Verhalten mit den geloggten Zuständen und Fehlermeldungen verglichen.
Um drittens zu prüfen, ob die Gesamtschnittstelle das beabsichtigte Verhalten zeigt, wurden externe Ereignisse simuliert und die Log-Meldungen beider Schnittstellenenden betrachtet.
Schließlich wurde eine Validierung des RBC-seitigen Schnittstellenendes durch Kopplung per TCP/IP mit dem Rail Simulation and Testing (RailSiTe)-Labor des Deutschen Zentrums für Luft- und Raumfahrt (DLR) vorgenommen, das den ESTW-Part übernahm.

5 Ausblick und Fazit

5.1 Zukünftiger Ausbau und Nutzung des Modells

Es wurde – abgesehen von der Multi-ESTW-Fähigkeit und der Fähigkeit zum Umgang mit einer ETCS/LZB-Doppelausrüstung – die gesamte Schnittstelle zwischen ESTW und RBC gemäß SCI-RBC in einem SysML-Modell inklusive eines ergänzenden Programmcodes umgesetzt. Durch Codegenerierung und Kompilierung erhält man daraus lauffähige Programme für beide Schnittstellenenden, die via TCP/IP kommunizieren können.
Um zu einem Modell eines RBC zu gelangen, sind als weitere Schritte das Hinzufügen der Schnittstelle zum Zug sowie das Hinzufügen von (ausgewählter) RBC-Kernfunktionalität in der Umsetzung erforderlich. In diesem Zuge werden auch weitere Tests des Modells notwendig werden.

the expected behaviour was compared to the states logged and error messages produced during execution.
Thirdly, in order to test whether the whole interface exhibits the intended behaviour, external events have been simulated and the logged messages of both interface parts have been examined.
Finally, a validation of the RBC interface part was conducted by means of coupling it via TCP/IP with the Rail Simulation and Testing (RailSiTe) laboratory of the German Aerospace Center (DLR), which acted as ILS.

5 Prospects and conclusion

5.1 Future extensions and use of the model

With the exception of the multi-ILS capability and the capability of handling lines equipped with ETCS and LZB (Linienzugbeeinflussung – German train control system) in parallel, the complete interface between ILS and RBC according to the SCI-RBC has been realised as a SysML model together with complementing program code. Applying code generation and compilation to it, executable programs are obtained for both interface parts, which communicate through TCP/IP.
In order to arrive at a complete RBC model, further steps such as adding the train interface and adding (selected) RBC core functions are currently in preparation. Related to this, new tests of the model will become necessary.
Accordingly, the integration into the rail operation simulation of the RailSiTe laboratory of DLR is planned to be extended (interface to trains simulated in the RailSiTe), so that the RBC can be fully used in the simulation.
In order to test whether an implemented RBC exhibits behaviour equal to that of the model, test cases are to be derived systematically from the model. This can be automated, at least in parts, with the help of suitable tools. Corresponding trials using parts of the model have already been conducted successfully and are to be continued in future. The use as a behavioural reference for a "real" RBC is always possible to the extent the functionality under test has already been modelled. A complete RBC model would, in addition to the aforementioned extensions, also require modelling of the interface between two RBC. Besides that, several practical questions such as the handling of manufacturer and country-specific RBC variants have to be solved before the successful use for the test of "real" RBC.

5.2 Discussion and conclusion

To construct a good model is a challenging task. In addition to familiarity with the system which is to be modelled, including its interfaces and the target language(s), knowledge of the modelling language, the modelling tools and the code generation is required. Design decisions and, connected to them, interpretations of the system specification, are much harder to come up with compared to the direct implementation in a programming language because it is more difficult to anticipate their consequences. In modelling practice there is a limited supply of applicable model elements – the more complex the data types in use (for example telegrams between ILS and RBC) and the more dynamic the system to be modelled (for example configurability of the RBC), the more complementing program code needs to be written and maintained in addition to the model. Other constraints regard limited available documentation or performance of tools.

Die bereits begonnene Einbindung in die Betriebssimulation im RailSiTe-Labor des DLR soll entsprechend ausgebaut werden (Schnittstelle zu im RailSiTe simulierten Zügen), so dass das RBC vollwertig in der Simulation nutzbar ist.
Um zu überprüfen, ob ein implementiertes RBC das gleiche Verhalten zeigt wie das Modell, sollen aus dem Modell systematisch Testfälle abgeleitet werden. Dies kann mit geeigneten Werkzeugen zumindest teilweise automatisiert werden. Entsprechende Versuche sind bereits mit Teilmodellen erfolgreich durchgeführt worden und sollen in Zukunft weitergeführt werden. Die Nutzung als Referenz-/Vergleichssystem für „echte" RBC ist jeweils soweit möglich, wie die zu testende Funktionalität bereits modelliert ist; für ein vollständiges RBC-Modell müsste zusätzlich zu den genannten Erweiterungen noch die Schnittstelle zwischen zwei RBC modelliert werden. Daneben sind bis zu einem erfolgreichen Einsatz für den Test „echter" RBC auch diverse praktische Fragen zu lösen, z. B. der Umgang mit hersteller- und ziellandspezifischen RBC-Varianten.

5.2 Diskussion und Fazit

Bis zu einem guten Modell ist es ein weiter Weg. Zusätzlich zum zu modellierenden System inklusive seiner Schnittstellen und der Zielsprache(n) sind Kenntnisse der Modellierungssprache, des Modellierungstools und der Codegenerierung nötig. Designentscheidungen und dafür nötige Interpretationen der Systemspezifikation fallen deutlich schwerer als etwa bei der direkten Implementierung in einer Programmiersprache, weil ihre Konsequenzen schwerer überschaubar sind. In der Modellierungspraxis

Undoubtedly the model-based approach does, however, have many advantages. The diagrams are much clearer and thus easier to maintain than program code, and the automatic code generation reduces the effort and error potential of programming. Those advantages became obvious during the modelling of the interface between ILS and RBC, for example in the shape of a very small effort for integration after finishing parts of the model (such as adding field elements). It can be stated that the RBC, or at least its interface with the ILS, fulfils the preconditions for a fruitful exploitation of the model-based approach. Large parts of the system behaviour can be captured in the form of state diagrams, and the model is suited for long-term use for several purposes.
Following the model-based approach for the RBC further, it might be a good idea to capture the railway infrastructure data in a suitable modelling language as well and to generate code from it for the configuration of the RBC. Concerning the choice of SysML for modelling the RBC as a software-oriented system, it could be said that the Unified Modelling Language (UML) as a software modelling language would have been equally suited. Those parts of SysML specifically tailored to system design needs did not play a role in the creation of a model from which code can be generated.
To summarise, the developed model of the RBC or, more precisely, its ILS interface can be expected to form a good basis for extensive, automatic and high-quality future tests of ETCS radio block centres and for demonstration of compliance with well-defined test criteria. ■

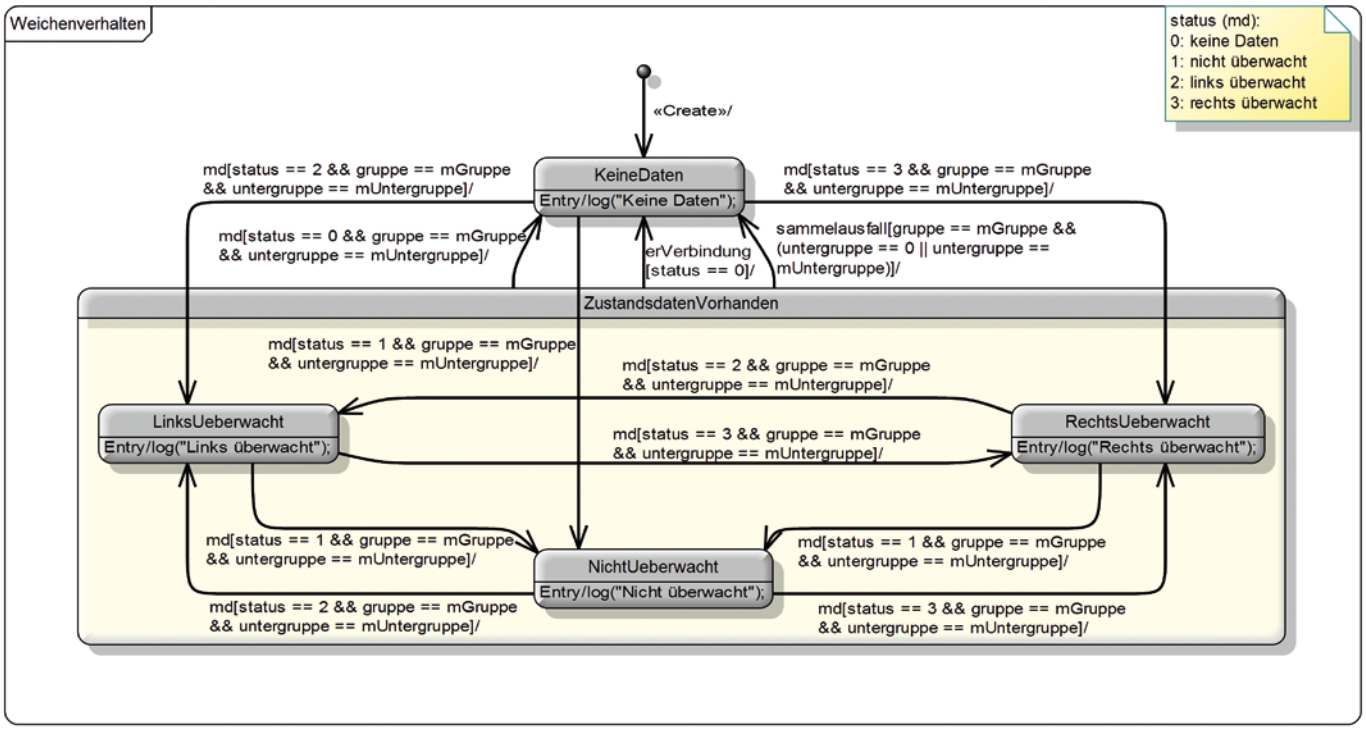


Bild 5: SysML-Zustandsdiagramm für die RBC-seitige logische Repräsentation einer Weiche
Fig. 5: SysML state diagram for the logical switch element of the RBC

stößt man auf einen beschränkten Vorrat an verwendbaren Modellelementen – je komplexer die verwendeten Datentypen (z. B. Telegramme zwischen ESTW und RBC) sind und je dynamischer das modellierte System (z. B. Projektierbarkeit des RBC) ist, desto mehr muss ergänzender Programmcode zum Modell geschrieben und mit ihm gewartet werden. Andere Hemmnisse betreffen beschränkte Dokumentation und Toolperformance.

Fraglos bringt der modellbasierte Ansatz jedoch viele Vorteile mit sich: die Diagramme sind deutlich übersichtlicher und damit besser wartbar als ein Programmcode; die automatische Codegenerierung reduziert Aufwand und Fehlerpotenzial des Programmierens. Diese Vorteile wurden während der Modellierung der Schnittstelle zwischen ESTW und RBC beispielsweise durch einen sehr geringen Integrationsaufwand nach der Fertigstellung von Teilen des Modells deutlich (z. B. hinzugefügtes Feldelement). Es lässt sich festhalten, dass das RBC bzw. seine Schnittstelle zum ESTW die Voraussetzungen, die einen modellbasierten Ansatz sinnhaft machen, erfüllt: ein großer Teil des Systemverhaltens lässt sich gut in Zustandsdiagrammen abbilden und es ist eine längerfristige Nutzung für verschiedene Zwecke beabsichtigt.

Denkt man den modellbasierten Ansatz für das RBC weiter, wäre es zukünftig vorstellbar, auch die Eisenbahninfrastrukturdaten in einer geeigneten Modellierungssprache zu erfassen und daraus Code für die Projektierung des RBC zu generieren. Was die Wahl von SysML für die Modellierung des RBC als softwarelastiges System angeht, lässt sich konstatieren, dass die Unified Modeling Language (UML) als Softwaremodellierungssprache ebenso gut geeignet gewesen wäre. Die spezifisch auf den Systementwurf ausgelegten Anteile der SysML haben für die Erstellung eines codegenerierungsfähigen Modells keine Rolle gespielt.

Insgesamt lässt das entstandene Modell für das RBC bzw. die Schnittstelle zwischen RBC und ESTW erwarten, dass damit zukünftig umfangreiche, automatische und qualitativ hochwertige Tests von ETCS-Streckenzentralen zur Erfüllung definierter Testkriterien durchgeführt werden können. ■

LITERATUR | LITERATURE

[1] Suwe, K.-H.: Proposal for a new type approval in signalling and telecommunications, Signal+Draht 7-8/2011, Eurailpress Hamburg
[2] EULYNX: Benefits of standardising interlocking interfaces, initiative "euLyNX", Project description, published on 06 February 2014 at eulynx.eu, accessed on 09 June 2016
[3] UNISIG: SUBSET-026 – System Requirements Specification, Version 3.4.0, date of issue: 12 May 2014
[4] UNISIG: SUBSET-039 - FIS for the RBC/RBC Handover, Version 3.1.0, date of issue: 09 May 2014
[5] DB Netz AG: Lastenheft BTSF3 – Betrieblich-technische Systemfunktionen für ETCS SRS Baseline 3, Version 1.4, date of issue: 03 March 2014
[6] DB Netze AG: SCI-RBC – FAS TAS TAV Schnittstelle ESTW-RBC V2, Baseline 0.19, date of issue: 06 June 2014
[7] Thales TS GmbH: Testspezifikation SCI-RBC, Version 01, date of issue: 21 February 2013

AUTOREN | AUTHORS

Daniel Schwencke
Verifikations- und Validierungsmethoden /
Verification and Validation Methods
E-Mail: daniel.schwencke@dlr.de

Hardi Hungar
Leiter der Gruppe Verifikations- und Validierungsmethoden /
Group Leader Verification and Validation Methods
E-Mail: hardi.hungar@dlr.de

alle /*all*:
Deutsches Zentrum für Luft- und Raumfahrt e.V.,
Institut für Verkehrssystemtechnik
Anschrift /Address: Lilienthalplatz 7, D-38108 Braunschweig