SIGNAL+DRAHT
Rail Signalling and Telecommunication

Sonderdruck aus
3/2011
www.eurailpress.de/sd

- High-quality interface specifications with SysML modelling

High-quality interface specifications with SysML modelling

Thomas Lauscher / Christian Fischer / Thorsten Hiebenthal

This article describes an approach that, starting with a coarse view of the system, leads to a detailed interface specification in clearly defined steps. The approach has been developed in the course of many technical discussions in collaboration with railway engineering experts of Deutsche Bahn AG. It is especially suitable for creating high-quality specifications in the complex environment of railway engineering. The procedure has become standard in the project NeuPro of DB Netz AG. Due to its clear structure, the approach is highly suitable for major projects with large teams of specifiers.

1 Introduction

Railway network operators are increasingly facing the challenge of integrating subsystems supplied by multiple manufacturers. Currently, many interfaces lack detailed specifications that manufacturers could use for guidance. As a result, entire systems are normally ordered from a single supplier.

Within the project NeuPro, DB Netz AG follows the objective of achieving higher flexibility by creating standardised interface specifications. Berner & Mattner has developed an interface specification approach in support of this goal. The approach, which is presented in this article, is based on modelling with SysML (Systems Modeling Language) [1]. It starts with a simplified view of the overall system and then follows a model-based and redundancy-free path with clearly defined steps to arrive at the final interface specification.

2 The challenge

The traditional approach for cooperation between railway operator and manufacturer provides for a clearly defined transition, which can be described in the V model according to EN 50126 between phases 4 and 5 (figure 1) of the RAMS life cycle (referred to as CENELEC phases below). The blue boxes indicate activities carried out by the railway operator, the yellow ones indicate the activities to be carried out by the manufacturer.

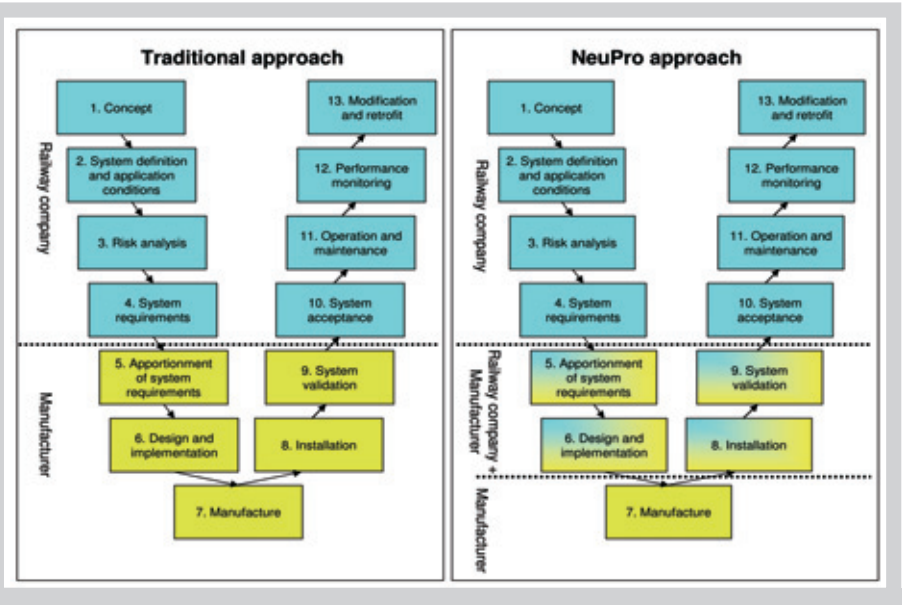

Figure 1: V model according to EN 50126 with old and new approach

The railway company uses specifications (CENELEC phase 4) to describe WHAT the system to be developed should provide. The manufacturer for his part develops specifications in several steps in CENELEC phases 5 and 6 describing the realization (HOW). Due to this classical task sharing, the manufacturer assumes the responsibility for the successful operation of the overall system comprising the IXL and the field elements. Consequently, the manufacturer has to ensure the interoperability of the subsystems. As a result, the railway operator can only procure the system as a whole.

In order to achieve higher flexibility in future, DB Netz AG has resolved to provide detailed and standardised interface specifications. This means that DB Netz AG will create interface specifications for those interfaces whose interoperability is to be ensured. This can only be achieved by taking over tasks which until now are left for the manufacturers to carry out. The right side of figure 1 shows this constellation, visualised by blue-yellow boxes.

As the left side of the V model illustrates, for example, considerations concerning architecture and distribution among individual subsystems (CENELEC phase 5). In some cases, predefinitions have to be set for CENELEC phase 6, e. g., regarding specification of electric currents and voltages. The right side of the V model reflects additional tasks the railway operator has to perform. These are tasks that so far have been taken over by manufacturers in CENELEC phases 8 and 9.

The interface specification is the most important document needed to enable interoperable systems from multiple manufacturers. An interface specification has to comprise the following requirements:

- Compilation and description of all functional processes to be covered by the interface,
- compilation of all non-functional requirements to be implemented by the interface, e.g., RAMS(S) requirements,

- detailing of all data to be transmitted via the interface,
- technical details to be considered to enable the two subsystems to be connected.

The requirements of these categories lead to a technical interface specification for all ISO/OSI layers.

It is the interface specification author's duty to provide a binding architecture and define in detail which tasks each subsystem covers. Finally, he has to describe the interfaces between the subsystems for all ISO/OSI layers in such detail that interoperability is ensured. These are tasks from the CENELEC phases 5 and 6 that railway operators have not yet been concerned with in this way.

A recipe-like approach is desirable to fulfil the requirements of writing interface specifications, which is quite a complex task. Such an approach would guide railway engineering experts through all the steps necessary to obtain the final interface specification. A procedure of this kind, currently being developed in the NeuPro project at DB Netz AG, is described below.

3 The solution path

The solution path is divided into two phases: modelling on the domain level and modelling on the technical level. The activity diagram shown in figure 2 graphically visualises the interface modelling process.

The domain level provides a functional, logical, and abstract view of the requirements, independent of particular solution concepts. Solution-related requirements, such as physical, electrical or software descriptions, are covered at the technical level. Tasks carried out at the domain level are referred to as the analysis phase in systems engineering.

The technical level uses a technical solution concept to implement the domain level requirements. In systems engineering, this phase is referred to as the design phase.

3.1 Activities at the domain level

At the beginning, it needs to be determined which of the subsystems of the overall signalling system are involved in the interface communication (referred to as interface end points below) and which of the subsystems are relevant for the interface to be specified. This is referred to as the interface context. SysML applies a block definition diagram for this purpose. It shows the static structure of elements (blocks in SysML) and their relations to each other (associations in SysML).

Subsequently, the interface definition requires determining which function is performed on which subsystem of the interface context and which of these functions require communication via the interface. In SysML, use cases reflect the functionality of a subsystem as services that a subsystem offers to another subsystem or to a person (referred to as actors in SysML). After defining the interface context, the interface-relevant use cases have to be identified and modelled in a use case diagram.

A use case consists of a sequence of actions executed in turn by the subsystem and the actor who is applying the use case. The activity diagram of SysML models this sequence of alternating actions between actor and subsystem. Which sub-function is to be executed on which subsystem and in what sequence is defined in detail.

Once the use case sequences have been worked out and the functional distribution among the subsystems has been fixed, the next step is a functional interface specification based on sequence diagrams. The necessary communication of functional information (commands and messages) across the interface is determined for standard workflows (successful process flow of a use case without faults) as well as for important deviant scenarios.

Sequence diagrams are suitable for visualizing the interaction of a number of selected communication scenarios. The goal is to obtain an interface specification that is as complete as possible, however. Therefore, a new diagram type now comes into play: Statecharts for subsystems of the interface end points can help cover and model all exception cases as well. In this way, the interface description is largely complete at the domain level. Lastly, statecharts offer execution and simulation options, facilitating further optimisation and requirement testing.

3.2 Activities at the technical level

Describing the concrete technical implementation forms the final step of the interface specification. At first, the ISO/OSI layers are described (see chapter 8.1). If industry protocols and standards such as Ethernet and TCP/IP are applied, it is sufficient to refer to their specification. Finally, the technical data telegrams are defined, including the description of their bytes and bits as well as their value ranges and meanings.

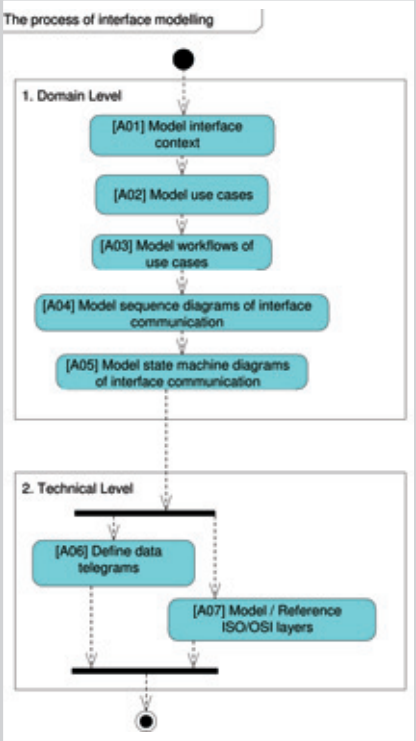

Figure 2: The interface modelling process as an activity diagram

The Berner & Mattner SysML process, SYSMOD and OOSEM

The process for developing interface specifications presented in this article is part of a comprehensive process developed by Berner & Mattner for modelling and developing railway engineering systems with SysML.

The overall process

- is based on the development processes (methodology) SYSMOD (Systems Modelling Process) and OOSEM (Object-Oriented Systems Engineering Method),
- implements the concept System of Systems (SoS),
- defines a model structure (implemented with SysML packages),
- supports the modelling of variants,
- integrates the phases of the RAMS life cycle [2],
- supports three presentation levels of railway engineering requirements (operational, domain, and technical levels)
- is easier to understand due to the limitation of the number of used SysML model elements, compulsory modelling guidelines and integration and preference of railway engineering terms.

The modelling of interface specifications described in this article can be integrated seamlessly and redundancy-free

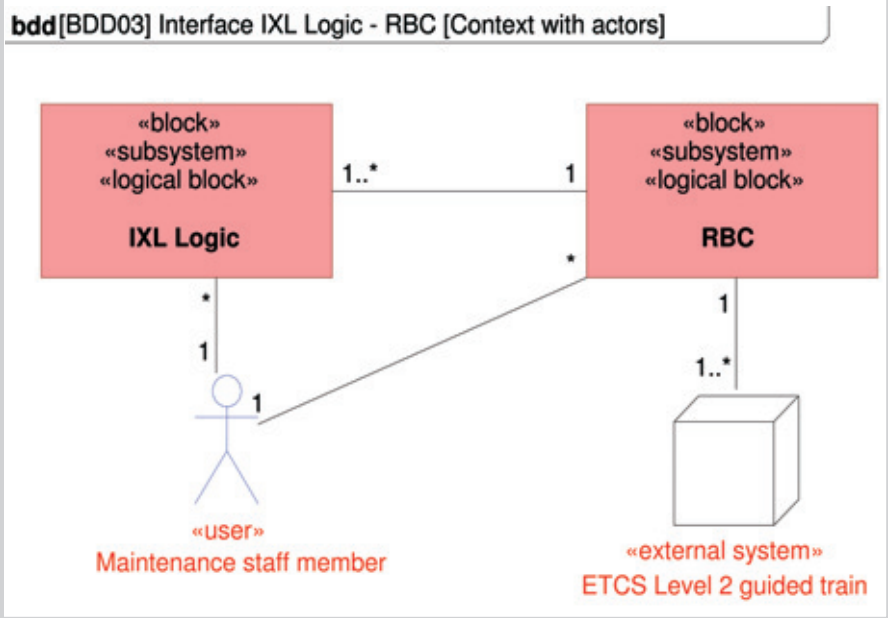

Figure 3: Block definition diagram "Interface context with actors" of the IXL logic – RBC interface

into the overall SysML signalling system model. Modelling of the operational level is not necessary for creating interface specifications and will not be discussed in any detail in this paper.

The underlying methodology, SYSMOD, has been described in the publication [3] by Tim Weilkiens, OOSEM in [4] by Sanford Friedenthal. Both contain a method for using SysML, from top level requirements through to a technical specification. The methodologies explain which SysML diagram type should be applied at what point of time in the process and in which way. Further, it is described how to link the different model elements in order to avoid redundancy and thus ensure traceability.

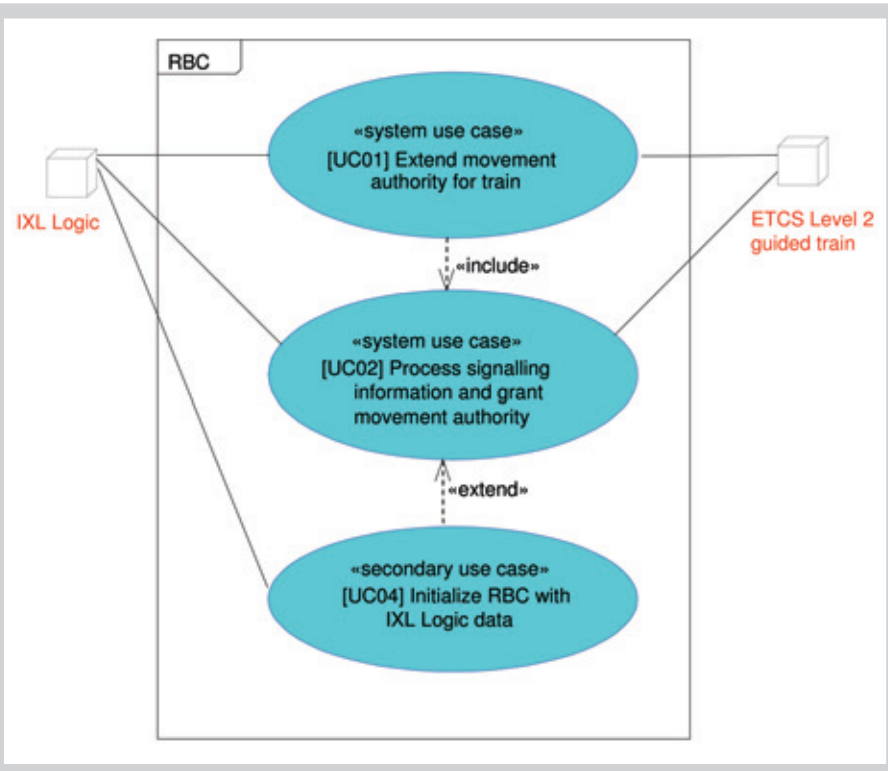

Figure 4: Use case diagram "Use cases of the RBC subsystem"

4 Interface context modelling

The interface context is modelled within a block definition diagram. The interface between the interlocking logic (IXL logic) and the Radio Block Centre (RBC) is shown in an exemplary diagram in figure 3. The displayed subsystems are the IXL logic, the RBC and two external actors – ETCS level 2 guided trains and a maintenance staff member.

The interface to be described is shown as an association between the IXL logic and the RBC. The RBC is linked to one or several trains running under supervision of ETCS level 2. The human actor "maintenance staff member" can communicate with the IXL logic as well as with the RBC.

5 Use case analysis

The use case analysis is indispensable for obtaining an interface specification even though the use case diagrams, as its direct results, do not always have to be included in the interface specification. In most cases, they are included in order to explain to the reader why certain functionalities require the use of the interface. In this case, the use case diagrams make for seamless traceability from higher-level use cases through to the last bit at the interface. If this seamless traceability does not appear to be necessary the use case diagrams will usually not be included in the interface specification.

The use case analysis is performed for each of the two interface end points in succession. This serves to determine which services each of the interface end points has to provide for the other and whether any given service necessitates communication via the interface. In this way, the system use cases are identified. System use cases represent the actual user goals of an actor at the interface end points. Exception situations and reused partial functionalities of use cases are modelled by secondary use cases.

As an example, figure 4 shows the parts of the RBC subsystem use cases that are linked to the RBC subsystem and therefore require interface communication and are relevant for the interface. The diagram contains two actors affected by the use cases. The secondary use case, "initialize RBC with IXL logic data", is linked to the primary case, "grant movement authority" using the "extend" relation. This shows that, under certain conditions, the initialization can be carried out within the use case UC04 (e. g., after connection losses). The use case "process signalling information and

grant movement authority“ is integrated via the ”include“ relation of use case UC01. This means that, at a certain point within the process of use case UC01, the use case UC02 is invoked as well.

6 Use case procedures – functional decomposition of the interface end points

After the use case analysis has been carried out it is clear which interface-relevant tasks both subsystems of the interface end points have to provide. However, neither has the process behind a use case been clarified yet, nor is it known how the individual process steps of this use case are distributed among the interface end points.

In order to determine this, an activity diagram is drawn for each interface-relevant use case illustrating the desired use case procedure. Partitions (swim lanes) represent the interface subsystems and define the functional correlation of the individual steps (referred to as actions in SysML) to the RBC and to the actors, respectively. This process step serves to distribute functions and is called functional decomposition.

The activity diagram is extended by additional details once the general workflows and the allocation to the subsystems have been defined. An action that is used in the use case activity diagram of a subsystem can be of one of the following types, according to the B&M definition:

- an invocation of another use case,
- a subsystem function,
- a subsystem action,
- a simple action.

A subsystem function is an essential function of the subsystem of a certain size describing the subsystem. Subsystem actions, on the other hand, are minor functions of a subsystem. The modeller has to make the distinction. The purpose of this differentiation is to prevent the list of the subsystem functions describing the subsystem from growing unnecessarily with minor and inessential functions.

A simple action (”grant movement authority“ in figure 5) cannot be re-used in other diagrams. The action has to be implemented as a subsystem action if reusability is desired and it is not a subsystem function.

An invocation of another use case is implemented as a call of the activity diagram of the called use case. This is marked by a fork symbol in the lower right corner in the action box.

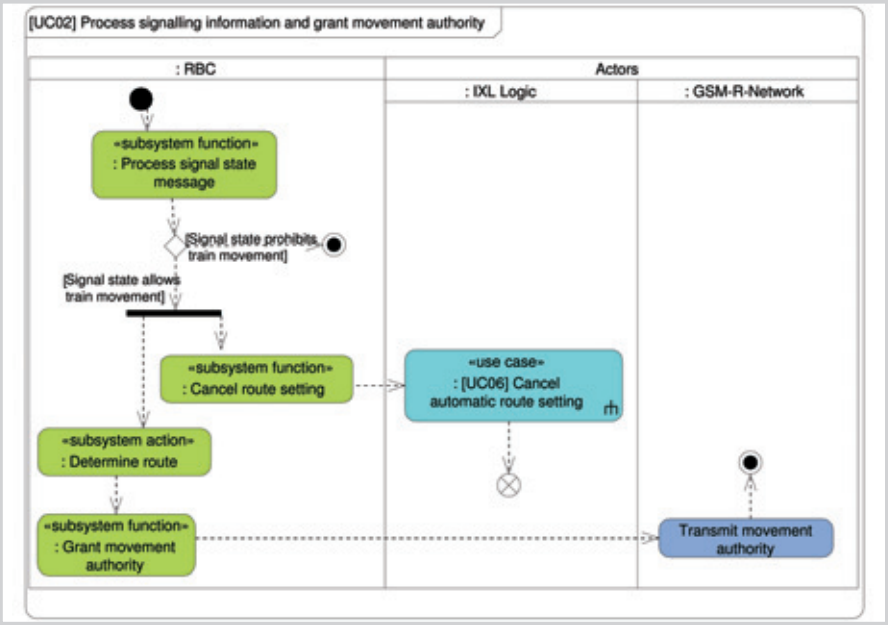

Figure 5: Activity diagram of the use case ”Process signalling information and grant movement authority“ of the RBC

A subsystem function and a subsystem action are realised as an invocation of a SysML operation of the respective subsystem block.

Using operations and invoked use cases in the described way ensures that the defined subsystem actions can be re-used in sequence diagrams and statecharts of subsequent analysis and design steps. Redundancy is avoided as well, leading to more consistent and accurate specifications.

This consistency and absence of redundancy can be seen in the following block definition diagram (see figure 6). By defining the subsystem operations within the activity diagrams, they have now become subsystem block operations. The operations can be re-used in all following analysis steps and displayed in various views.

7 Communication Modelling

So far, the characteristics of the system consisting of the two interface end points have been modelled. The communication across the interface is described in the next step. This includes the specific procedures at the interface, which are visualized by sequence diagrams.

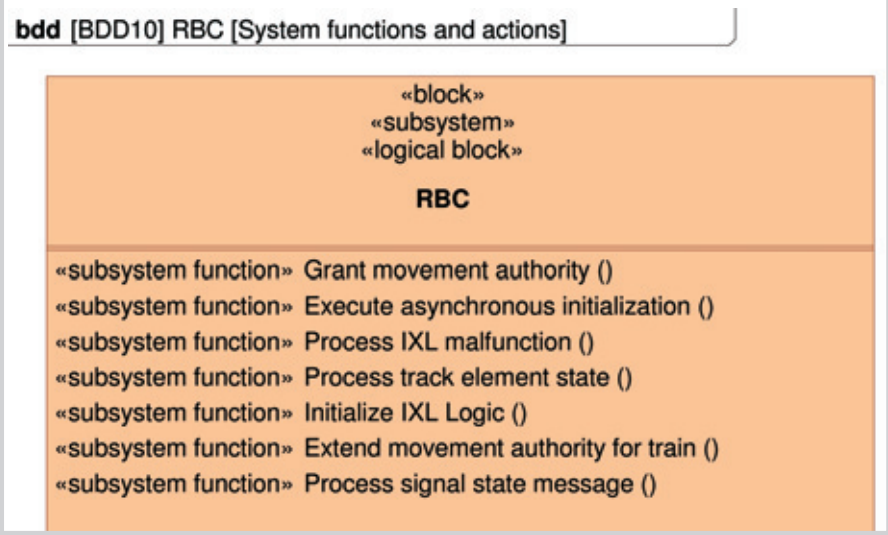

Figure 6: Block definition diagram with system functions and system actions of the RBC

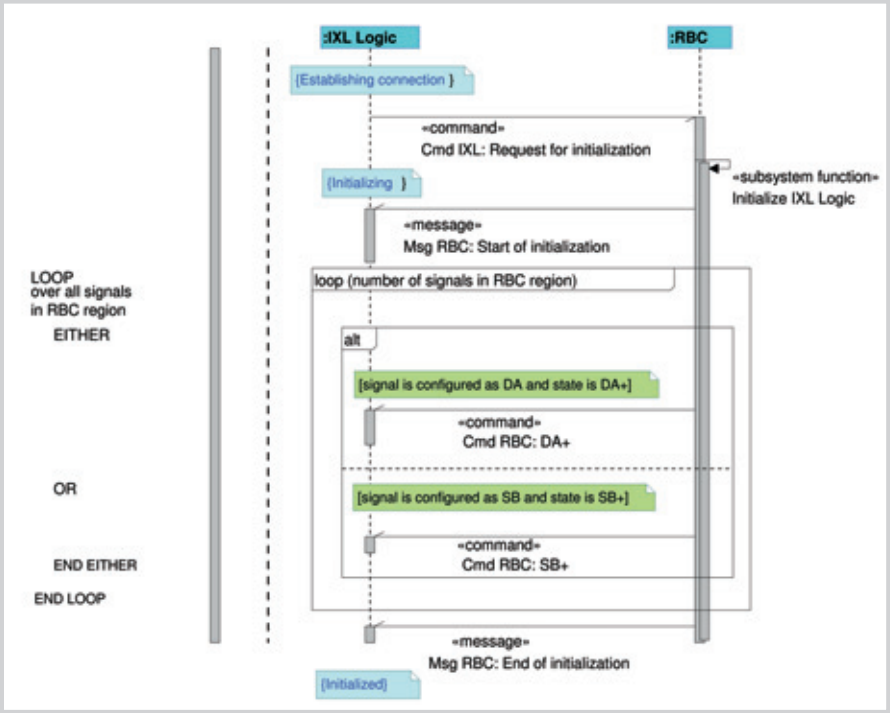

Figure 7: Sequence diagram of an example scenario, IXL logic initialization

Furthermore, the communication modelling includes the response of both subsystems to external stimuli, depending on the current state of the subsystems. These details are imaged in statecharts.

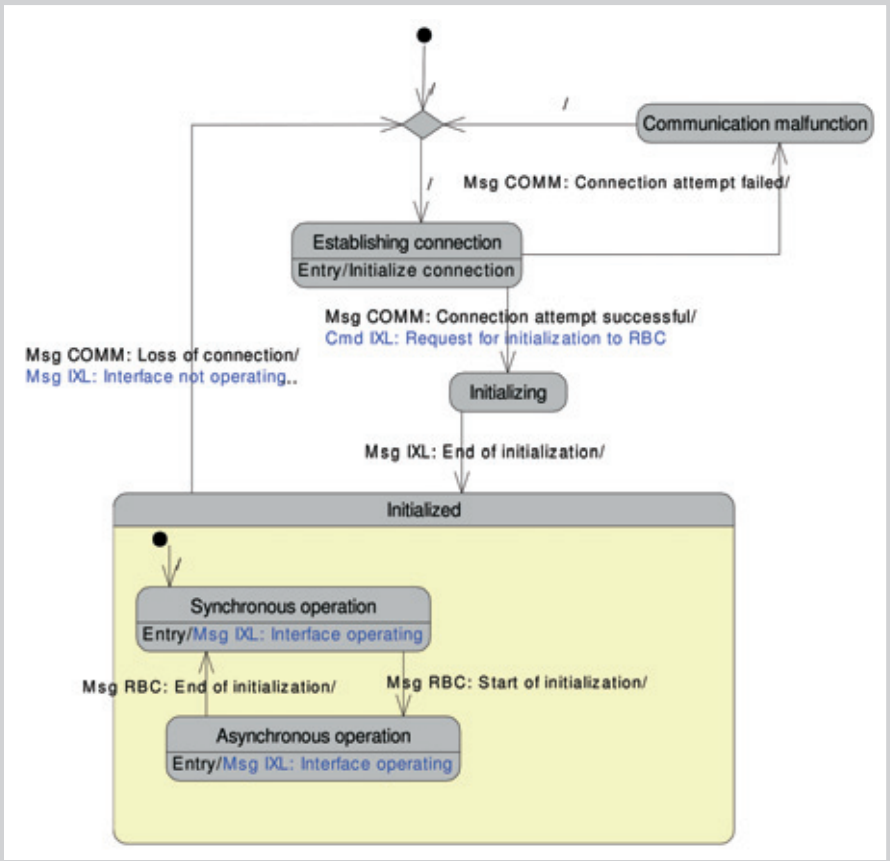

Figure 8: Statechart: states of an interface area of the IXL logic subsystem

7.1 Communication modelling – sequence diagrams

First, sequence diagrams are used to describe the normal behaviour at the interface. Each sequence diagram covers exactly one scenario (sequence of actions within a use case that has to be executed under certain conditions). Though sequence diagrams do allow optional procedures, these tend to make them more difficult to read. Therefore, optional procedures are used very sparsely in the presented approach. Thus, a series of sequence diagrams is generally needed to illustrate all relevant functionalities that have to be supported by the interface.

An example of a sequence diagram is shown in figure 7, describing the functional information objects that have to be sent via the interface to accomplish the initialization of the IXL, triggered by the RBC. The information objects can be divided into commands and messages. They are used throughout, starting with sequence diagrams and after that by state modelling. Finally, they are defined by concrete telegram content on the technical level. The benefit of this approach is that an individual piece of information that is transmitted via the interface always carries the same name – whatever diagram type it is used for or whichever modeller has modelled it. This enables modelling without redundancy, which, in turn, facilitates re-usability of model elements.

The use of state invariants (indicated by light blue rectangles in figure 7) is another characteristic of sequence diagrams. States of the statecharts, covered in the following chapter, are used for these invariants. They facilitate an extended understanding of the system context. A relationship to the subsystem functions described in chapter 6 is established by modelling operation calls that are executed in response to received commands and messages.

7.2 Communication modelling – statecharts

Statechart modelling describes how SysML blocks react to stimuli by certain events, depending on the current state of these blocks. In general, they can represent overall systems, subsystems, or logical components. In order to create interface specifications statecharts for subsystems of both interface end points need to be drawn

As an example, figure 8 shows the statechart of the interface area of the IXL logic. The subsystem interface area

represents the part of the IXL logic that contains all logic required for communicating with the RBC. The first thing that catches the eye in the diagram is that commands and messages are used as in the sequence diagrams. They occur as incoming events the system is reacting to (e. g., the message, “Msg Comm: connection established“). They can, however, also be outgoing events that are sent to the respective other system during a state transition (e. g., the command, “Cmd IXL: Request for initialization to RBC“, which is sent during the transition into the state ”Initializing”).

When state modelling interfaces, it is important not to try to model the complete internal behaviour of both interface end points. The objective rather must be to identify the relevant parts of the interfaces and to design their state models. Interface-relevant system states are states either receiving or sending information objects via the interface.

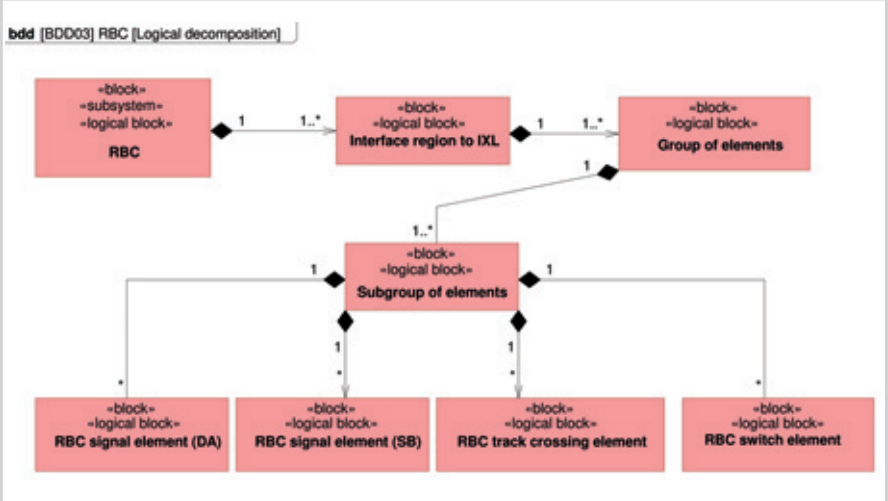

Figure 9: Block definition diagram, logical decomposition of the RBC

7.3 Logical subsystem decomposition

In the previous chapter, the statechart did not cover the entire subsystem IXL logic. In fact, it only contained a part of the IXL logic, namely, an interface area. The fact that an IXL logic contains several interface areas is the reason why the interface communication cannot be located directly in the statechart of the IXL logic. Therefore, parts of the subsystem have to be defined within the interface specification to some extent as well. Underneath a component, the interface communication can now be modelled as a statechart for a single component.

Additionally, for other reasons, it can be necessary to decompose the subsystem into logical components, to allocate them to functional groups or to draw the statecharts in a more clear, legible, and structured way, for instance. In figure 9, the RBC subsystem is split up into logical components.

It should taken into account that the logical decomposition does not have to be complete. Concerning the specification of interfaces, this would create additional, needless effort. It is rather recommended to draw only those logical components that are needed for the interface specification

8 Technical implementation guidelines

After the statechart modelling has been finalised, the description of the affected subsystems’ structure and interface-relevant behaviour is complete. Concrete guidelines for the technical implementation are still missing, however. Supplier-independent interoperability of an interface can only be achieved by prescribing important technical details mandatorily. The railway company must provide technical constraints that were formerly in the manufacturer’s area of responsibility. These constraints are necessary to prevent different manufacturers developing incompatible solutions.

Byte-Nr	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	Telegram number 1234 (0x4D2)							
3	Transaction number							
5	L = number of characters in Text							
6	Text (ASCII, max. 40 Bytes)							
6+L	Control byte							
7+L	Security appendix							

Telegram information:

Byte 1	Telegram number (binary coded). Designates the type of telegram.
Byte 3	Transaction number (binary coded). This number identifies a control action by a user and its corresponding acknowledgement.
Byte 5	Number of characters in Text (binary coded)
Byte 6	Text (ASCII, max. 40 Bytes), no trailing 0 needed.
Byte 6+L	Control byte (binary coded) 0 = delete text from view 1 = show text
Byte 7+L	Security appendix – a CRC over the preceding telegram content.

Telegram usages:
1. command “Send Text”
Control byte = 1
2. command “Delete Text”
Control byte = 0

Figure 10: Telegram structure, description of telegram fields, and usage of telegrams

N°	OSI layer	Tasks	Protocol examples
7	Application	- Provide applications access to network	HTTP, FTP, HTTPS, SMTP, LDAP
6	Presentation	- Data conversion into standard format for transmission - Data compression - Data encryption	
5	Session	- Session management (Start/end of communication, reintegration) - Handling of disconnections	
4	Transport	- Logical addressing of processes at hosts - Error prevention and error correction process - Congestion control	TCP, UDP
3	Network	- Logical addressing of hosts - Routing (e.g. via routing tables) - Flow control	IP, ICMP
2	Data Link	- Physical addressing of hosts - Ensuring of reliable transmission - Definition of the arbitration method for bus communication	Ethernet, Token Ring, FDDI
1	Physical	- Transfer medium/line - Line code/bit coding - Serial or parallel transfer - Symmetrical or non-symmetrical transmission	

Table 1: The ISO/OSI layers of the reference model

Signalling systems almost exclusively use telegram-based data interfaces. Therefore, how to define interface telegrams leaving no room for interpretation is illustrated in the example below.

As an example, the upper part of figure 10 shows a structure of a data telegram, that is, its byte and bit structure. This is followed by a description of all telegram fields in the middle part. Finally, the ”telegram usage“ is defined for every telegram instance in the lower part.

In this context, the term telegram usage refers to a specific usage of a telegram as a command or a message. The telegram usage definition describes the content of all fields that represent the specific usage as a command or a message. In the example shown above, it only refers to the control byte because this differentiates between the two telegram usages, ”send text“ and ”delete text“. Fields are not mentioned here if their content is variable within the given telegram usage, the text to be sent, for example.

Each telegram usage is given a unique name within the context of the interface. The name is used in the sequence diagrams and statecharts, helping to determine exactly how a telegram should be composed for any given task at any time.

8.1 ISO/OSI layers below the application level

The interface requirements developed at the domain level describe the interface at the functional level. Besides these functional requirements, an interface also has to implement non-functional requirements such as RAM(S) requirements. Additionally, decisions have to be made regarding the actual technical implementation at the technical level. The ISO/OSI reference model [5] is widely used for interfaces (see table 1).

The developed application procedures reside on layer 7, the application layer.

Due to its layered architecture, the ISO/OSI model facilitates a clear separation of network communications responsibilities and tasks. Moreover, it helps to define and apply industry standards for individual layers and to make individual layers replaceable. One approach could be, for example, to apply the network protocol TCP/IP for layers 3-4 and to use Industrial Ethernet for layers 1-2.

To meet safety requirements, an intermediate safety layer between the transportation layer (4) and the presentation layer (6) could be specified and developed by modelling.

SysML facilitates the presentation of these layers and their interactions with block definition diagrams and internal block diagrams.

LITERATURE

[1] OMG Systems Modeling Language (OMG SysML) Specification, Internet: http://www.omg.org/technology/documents/domain_spec_catalog.htm#OMGSysML, OMG

[2] EN 50126:1999 – Spezifikation und Nachweis der Zuverlässigkeit, Verfügbarkeit, Instandhaltbarkeit und Sicherheit (RAMS), VDE

[3] Weilkiens, T.: Systems Engineering mit SysML/UML, dpunkt Verlag, 2. Auflage 2008

[4] Friedenthal, S.: A Practical Guide to SysML – The Systems Modelling Language, Elsevier, 2008

[5] Zimmermann, H.: OSI Reference Model-The ISO Model of Architecture for Open Systems Interconnection, 1980

Berner & Mattner Systemtechnik GmbH
Erwin-von-Kreibig-Str. 3
80807 München
Tel.: +49 (0)89 608090-0
Fax: +49 (0)89 6098182
info@berner-mattner.com
www.berner-mattner.com

The authors

Dipl.-Ing. Thorsten Hiebenthal
Head of Transportation
Berner & Mattner Systemtechnik
Address: Erwin-von-Kreibig-Straße 3, D-80807 München
E-Mail: Thorsten.Hiebenthal@berner-mattner.com

Dipl.-Ing. Thomas Lauscher
Senior Systems Engineer
Berner & Mattner Systemtechnik
Address: Erwin-von-Kreibig-Straße 3, D-80807 München
E-Mail: Thomas.Lauscher@berner-mattner.com

Dipl.-Inf. Univ. Christian Fischer
Software Engineer
Berner & Mattner Systemtechnik
Address: Erwin-von-Kreibig-Straße 3, D-80807 München
E-Mail: Christian.Fischer@berner-mattner.com