Modelling and Simulation of Train Control Systems using Petri Nets

Michael Meyer zu Hörste
Institut für Regelungs- und Automatisierungstechnik
Langer Kamp 8
38116 Braunschweig
Tel.: +49 531 391 33 31
Fax.: +49 531 391 51 97
E-Mail: M.Meyer-zu-Hoerste@tu-bs.de
www.ifra.ing.tu-bs.de

Abstract

A formal model was prepared on behalf of the German railways (Deutsche Bahn AG) using the informal ERTMS/ETCS specification as a basis. Proceeding from the existing models of the system a model of the system design was developed which is to provide a universal means of description for all the phases. It was decided to use Petri nets as a means of description for this procedure, as they permit universal application, the use of different methods and formal analysis. The method developed is an integrated event and data-oriented approach, which shows the different aspects of the system on their own net levels. The model comprises three sub-models with a model of the environment developed next to the onboard and trackside systems. This environment model covers all the additional systems connected through the system interfaces, examples of which are interlocking or regulation. Starting from a net representing the system context, the process of the onboard and trackside sub-systems was modelled. Here, the different operations and processes are visualised in the form of scenarios, which in turn have access to additional refinements representing specific functions. System modelling was supported by the tool Design/CPN selected after careful evaluation. ERTMS/ETCS system modelling was taken to a point permitting partial model simulation. On the basis of these models, additional options of the spiral model of the system design now appear: the train and RBC models may expand into specific visualisations, the algorithms can be further refined and compared, the models can be used for different kinds of tests and also for purposes of system quality assurance, which may go as far as furnishing proof of safety standards. Additional phases of system development may now be elaborated on the basis of the spiral model. Further investigations will have to aim at deriving from a model both the source code and the executable code.

1 Introduction: ERTMS/ETCS modelling

Implementation has to start from a situation where a host of documents have been prepared for specification purposes. Characteristics of these documents are that they specify different aspects, are based on each other, and have different objectives [1,2,3]. From this follows that there are separate documents for specific sub-functions and sub-systems which specify a certain segment of the requirements made on the system, a phenomenon which is most apparent with the central systems which immediately adjoin the interoperable interfaces. The documents can be seen as forming a specification network interlinked by references and cross-references. All this means that, in order to finalise specification work and set about implementation, the specification has to be proved to be fully consistent, and system operability has to be verified; it should also be checked that the system suits the operational conditions of the different railway operators and countries. In the past, a multitude of highly diverse methods and means of description, employing specific computer tools or even manual operations, were used to meet these requirements.

Deutsche Bahn AG and the Institute of Control and Automation Engineering set about formal modelling work on the basis of Petri nets. The intention was to demonstrate that a universal means of description can be used in all the phases of system design. The system model developed has the following features: the overall system was subdivided into the sub-models onboard system, trackside system (Radio Block Centre - RBC) and environment. The latter covers all the other systems associated with the former two.

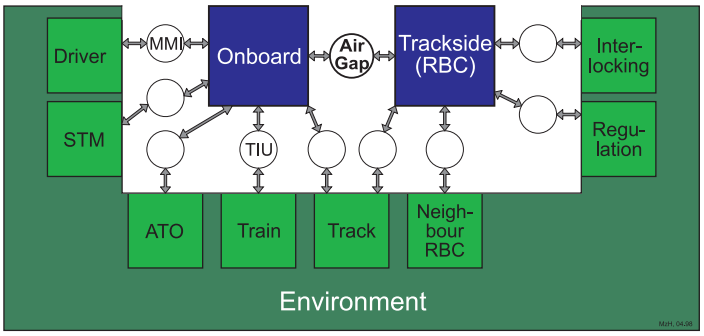

Fig. 1: System architecture

One central aspect was the air gap and the balise interface, which were to be modelled with as much detail as possible and in compliance with the System Requirements Specification [1].

2 Definitions

2.1 The term 'model'

A model can in a general way be characterised by three properties: a model represents part of the reality, and of that reality those features that are of relevance for the immediate purpose of the model; other features that are without relevance for the purpose of the model are only presented in a reduced form. The model serves a specific purpose; this characteristic of a model is referred to as pragmatics. A model can serve to describe certain conditions, it can provide insights, or it can replace a real system in a specific limited way.

2.2 Verification, Validation and Test

Within the Systems Life Cycle it is very important to distinguish between the terms Validation, Verification and Test. All of these three methods are approaches to ensure the quality of the system in a step of the systems development, but have different objectives and bring specific results.

Verification
Verification is an examination, whether a systems fulfils specified requirements and is a proof that a transformation between two representations of the system is faultless. Verification can not prove that the specified requirements are themselves correct, this should be checked with a validation. [4]

Verification can be performed in a formal way, but needs a formal description of the system which is accessible for analysis methods.

Validation
The Task of the Validation is an examination, whether the system is suitable for the intended use. The Validation needs a specification of the environment within the system should be used and the result is always related to this environment. Validation outside of the defined environment is critical, because the assumptions about the systems environment are not valid any more. [4]

In the context of the development of train control systems a validation is the check, whether a system is suitable for the operational requirements of the railway operator. Such a check can not be performed in a formal way, only the knowledge of the operating personnel can give information about the suitability for real situations.

Test
A test is the procedure that the behaviour of an implementation the system with selected representative test-data. A complete test i.e. with all possible Input-data is in general impossible, because of the huge number of input-data and parallel paths of working. [4]

In railway environment there exist a lot of different operational scenarios and specific situations which cause a huge number of test cases.

2.3 Means of description, method and tool

The GMA sub-committee 1.8.1 Standardisierte Beschreibungsmittel in der Automatisierungstechnik (Standardised means of description in automation engineering) provides the following definitions for the terms 'means of description', 'method' and 'tool' [5]:

> ***Means of description:***
> *A means of description describes graphically certain conditions for visual perception and storing. Means of description are alphanumeric signs, symbols and other graphic elements of representation (semiotics,) and also conventions on how these can be combined (syntax). Assigned to the different elements of representation, their possible combinations and allocations are specific conditions and concepts from a certain context, which may be specified in a more or less detailed and formal way (semantics).*

The following distinctions are made:

> ***Formal means of description:***
> *Has a mathematical basis and a defined and complete syntax.*
> ***Semi-formal means of description:***
> *Has a defined and complete syntax, not, however, a mathematical basis.*
> ***Informal means of description:***
> *Also possesses the characteristics of a means of description (semiotics, syntax, semantics), these are, however, not always complete.*
> *Example: natural language.*
> *For formal and semi-formal means of description there is an unambiguous interpretation.*

The means of description used in modelling ERTMS/ETCS are coloured and hierarchic Petri nets, since the aim is to examine whether these offer the possibility of using one uniform means of description for the entire development cycle, starting with the specification through to implementation. Above and beyond that, Petri nets provide the required capacity that allows different methods to be used during one single phase of the development cycle and also phase-specific methods [6].

> ***Method:***
> *A method is a procedure, systematic both in terms of the point in question and the purpose, following a set of principles and designed to produce insights and practical results.*

In the project presented here, an integrated method was used, which has both a data and event-related orientation. It provides for visualisation of the sub-system processes, but also of the operational processes in the form of scenarios, and of the functions in individual nets.

> ***Tool:***
> *Designed to assist man in or during the production of results. Today, the term 'tool' is normally understood to mean 'realised by computer systems (hardware / software)'.*

Following comparative investigations and studies, Design/CPN was selected as the tool for this project. The decision was not least taken because of the fact that it provides for a reachability analysis [6,7].

3 System development using formal methods

Starting from a system life cycle - the so-called waterfall model [8] - a specific way of representing the phases of system development was created for modelling. The phases are shown in Fig 2. A distinction is made between the specification proper and additional models that serve purposes of documentation and visualisation. This does not only follow from the specification itself, which extends from the *idea* to the *model with algorithms*, but also from the requirement of documenting, elucidating or visualising properties of the intended system. During the functional specification phase, visualisation of the comprehensive operational behaviour is of prime interest, while during the following phase the interest is more on the specific behaviour in defined operational processes, which can be illustrated by simulating so-called scenarios. The different individual scenario simulations can also be integrated into an overall simulation to represent the defined operational processes. During the phase of system

specification, the entire system behaviour can already be represented as such and is used to describe and visualise the intended behaviour.

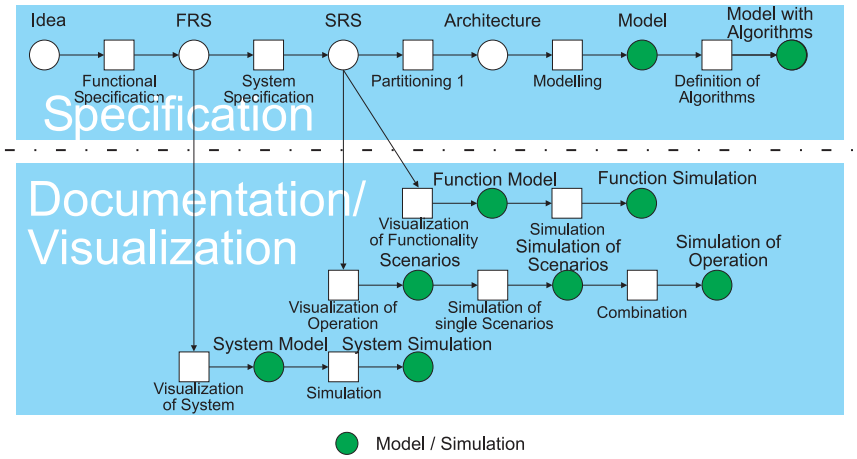

Fig. 2: The waterfall model

Fig. 2 is to give a general idea of the two different strands of modelling. While the top section shows the sequence of development phases, starting with the system presentation through to a detailed specification in the form of a model, including the required algorithms, the bottom section comprises the sub-functions of visualisation and documentation, which are conceivable as elements accompanying development. It makes sense to differentiate between specifying and visualising models, as the orientation of the model also has an influence on the way of how modelling proceeds. Models used for specification inevitably comprise a host of details which make it difficult to quickly grasp the more general context. It is for this reason logic to provide models for visualisation of specific conditions, which have specifically been abstracted for this particular purpose.

In compliance with the spiral models [9], this procedure was extended to comprise the aspects of quality assurance.

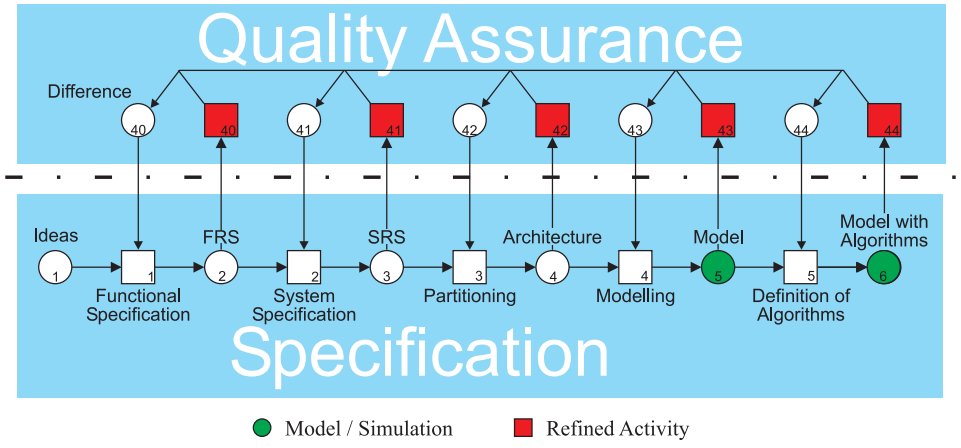

Fig. 3: Extended model of system development

With each step, the results are checked for compliance with the targets. Should this check-up produce any differences, it is inevitable to return to an earlier phase. In this context, the three

following methods of quality assurance can be applied to the models: test, validation by means of simulations, and verification using the methods of formal analysis.

Refining the squares in Fig. 3, this can be depicted in the following way:

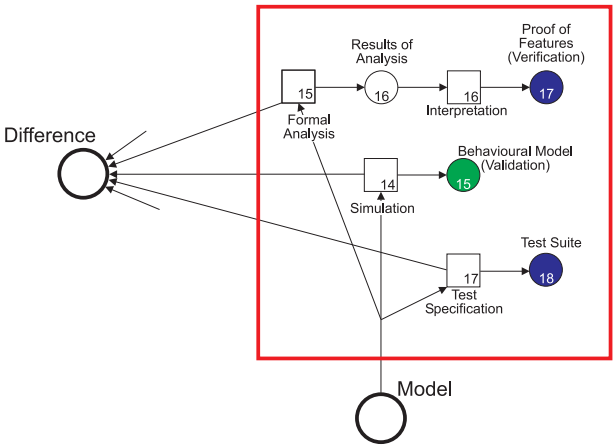

Fig. 4: Quality assurance on the basis of models

Tests can, in particular, be applied to implementations, and they can test agreement of the implementation in respect of specific system properties using a test reference. In this case, a formal model could serve as a test reference. Validation on the basis of simulations is appropriate, in particular, during the early phases of system development, as at this stage the system has as yet not been sufficiently defined for formal analysis. During the phases of system development, i.e. when formal models can be used, formal analysis is a suitable instrument for proving specific properties. A model can be subjected to specific analyses which verify specific properties of the model in an abstract manner by calculation [10]. Initially, the findings do not have a direct relationship with the specified system. To be able to be translated into specific system properties, the abstract results have to be interpreted.

4 Model structuring

4.1 Principles of net structure

For ERTMS/ETCS modelling it was decided to visualise system context, sub-system process, operational processes in the form of scenarios and functions in an integrated manner. For this purpose, nets are decomposed to reflect these four aspects. This can be represented in the following general form:

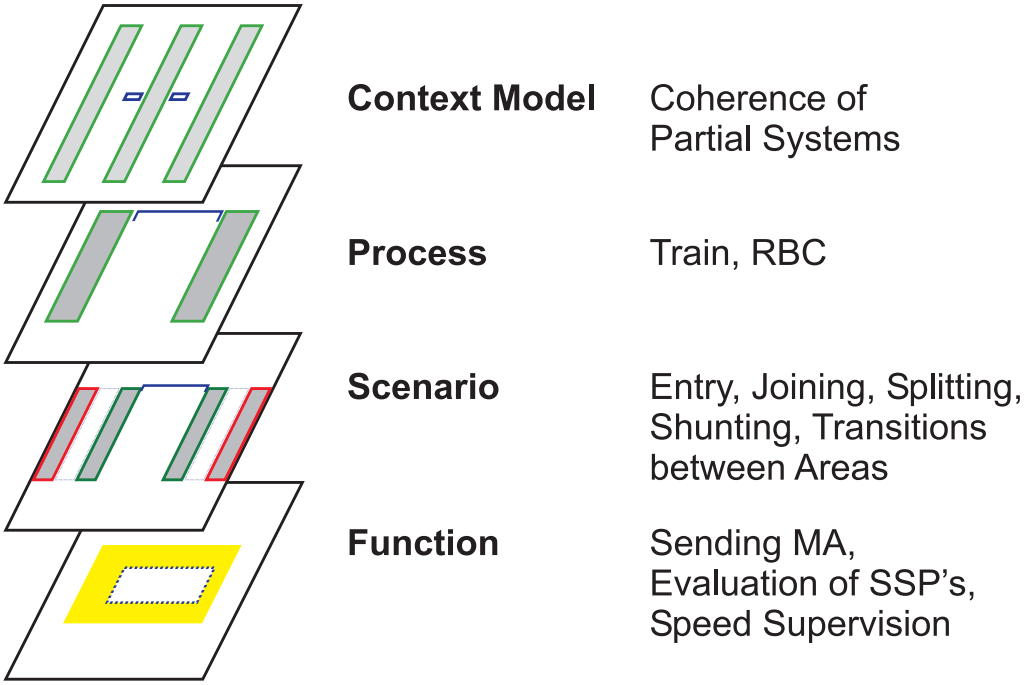

Fig. 5: The different types of nets within the net structure

All of these four different types of nets are defined by a specific pattern, which are explained in the following.

4.2 Overall model / context net

The system context is depicted on the uppermost level. This net comprises the two modelled sub-systems onboard system and RBC. All the other sub-systems are comprised in the environment, they may, however, be further refined during a later phase of modelling work.

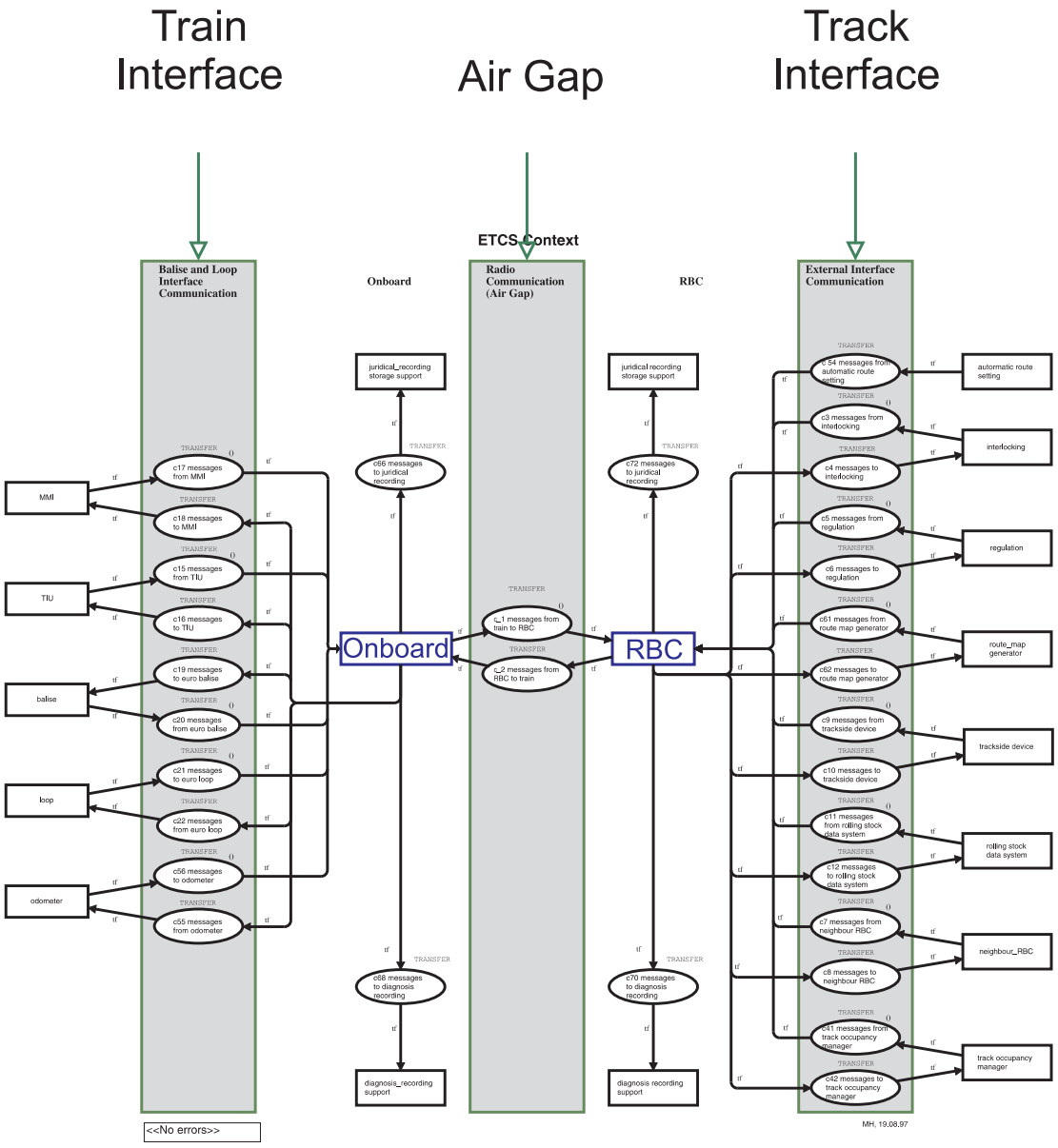

Fig. 6: The net of the context level

This net corresponds to the formal representation of the system architecture in Fig. 1. On this level, all the interfaces are defined as uni-directional channels.

4.3 Nets on the process level

The next level is formed by the nets of process visualisation. This level defines what scenarios can be passed in what sequence.

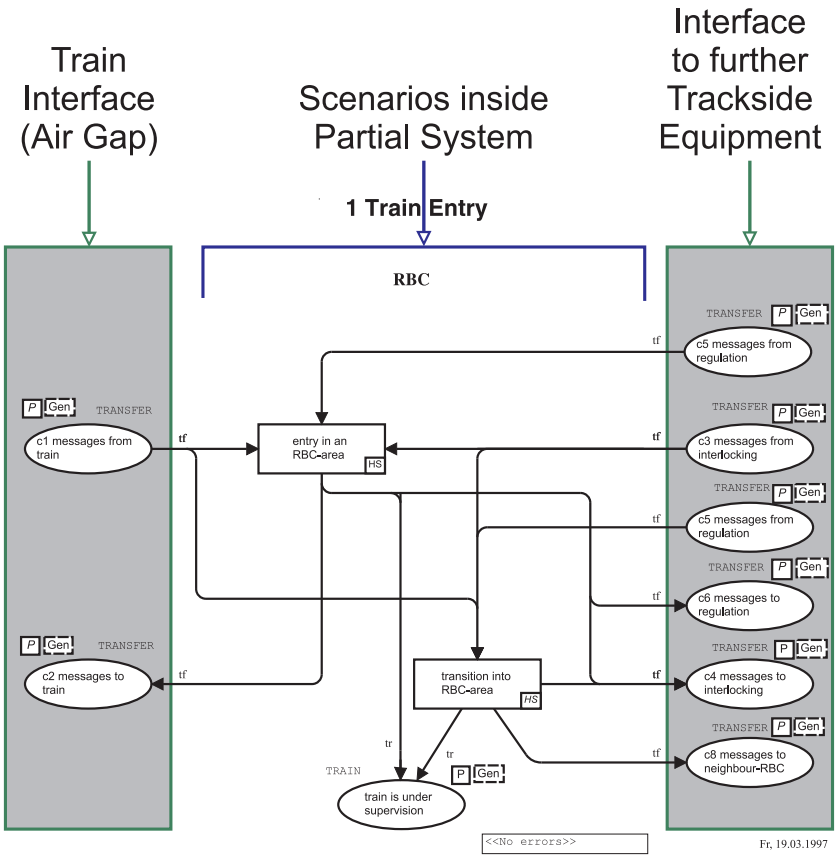

Fig. 7: Example of nets on the process level

In a central position are the transitions and places that provide additional application logic. To the right and to the left are two dark-grey boxes, which accommodate the interfaces to the outside. Messages sent to a receiver or received by a sender are combined in one place. At the train end the right-hand box, and at the RBC end the left-hand box, is reserved for communications between train and control system.

4.4 Scenario nets

The different scenarios are refined with two more aspects. The interfaces are disintegrated until individual messages are singled out. For each message there is a transition, referred to a driver, which is responsible for translating the message from the general data type *"message"* into the required context specific type. In this context, messages are defined strictly in compliance with the definitions in chapters 7 and 8 of the System Requirements Specification, version 4 [1]. In the nets themselves, the sequence of events is shown. The possibility of parallel event sequences as a function of specific initial situations and stimulations is given due to consideration. Used as a basis here is SRS, version 4.0 [1] and the scenarios, version 2.0 [2].

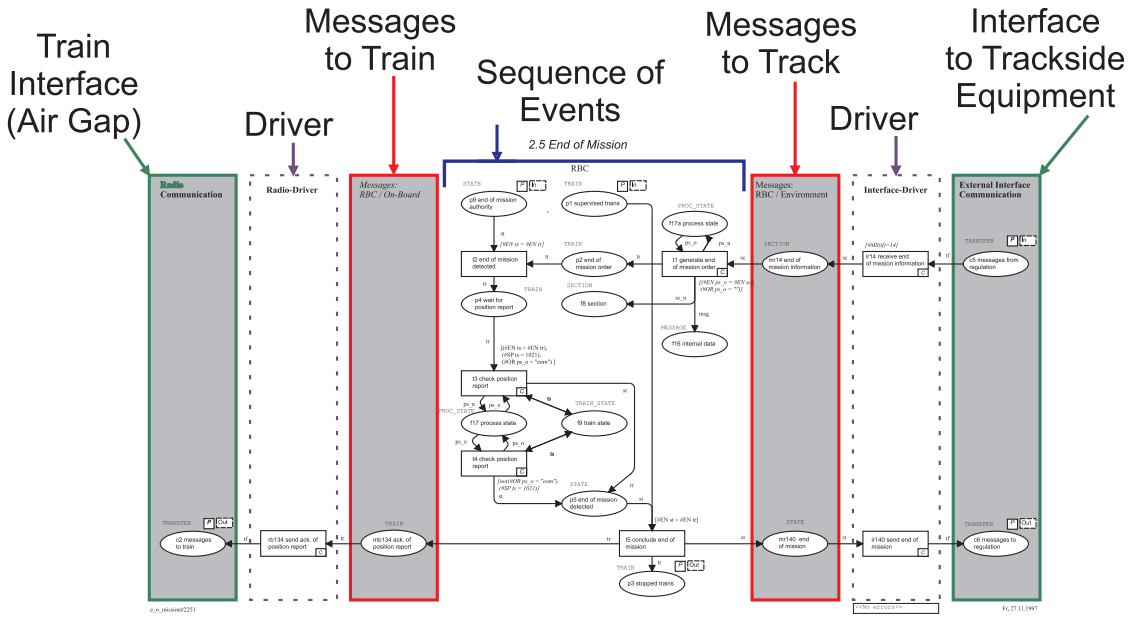

Fig. 8: Example of a scenario net

These nets comprise three areas: the application logics is again located in the centre (1), while the two boxes with the interfaces are located at the extreme left and right (2, 3). Next to the interfaces is a box designated radio driver (5) and interface driver (4), respectively. All transitions in these boxes start with 'send' or 'receive'; they comprise the receive or send logics for a specific message. This is why between these boxes and the application logics there are one or several more dark-grey boxes (6, 7), providing one place for each individual message. The name of this box (these boxes) starts with "messages".

4.5 Functional nets

The functions, if explicitly modelled, form a net level of their own, and are visualised in the following way.

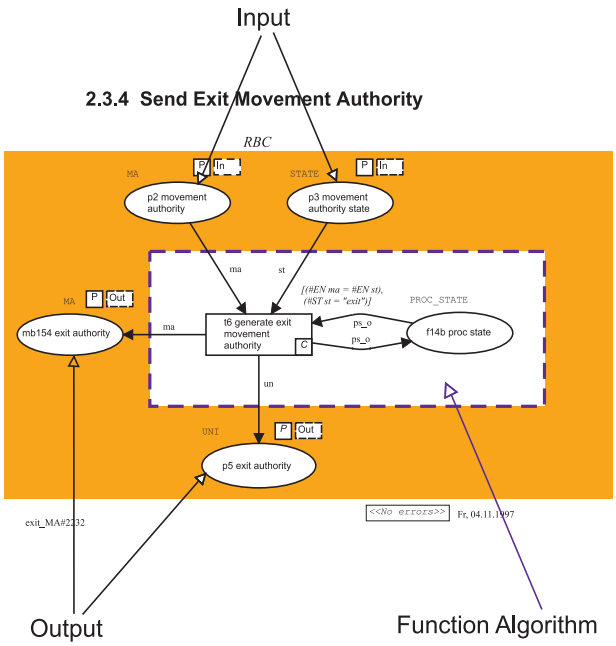

Fig. 9: Example of a functional net

The nets on this level are much simpler in structure: they only show the transition refinements. This is why they do not have any interfaces with external systems, even if the transition itself has such interfaces. Communication is safeguarded by the higher (scenario) level net. The net logics is accommodated in a box delimited by a broken line (1); the inputs and outputs of the transitions of the higher-level net are outside this box (2).

5 Results and model performance

5.1 Analysis

Regarding formal analysis methods of Coloured Petri nets, we did some work on how state space analysis could be applied in a sensible way to our modelling approach and what the results can be used for.

First, with our approach analysis capabilities are restricted by the fact that the component models are open. State space analysis of a single component model can be done for specific scenarios given as sequences of stimuli to the model. An analysis of the complete behaviour of an isolated component model is not possible with our modelling approach.

Even if there existed an environment model serving as a closure to the isolated model under consideration, the level of detail would not allow a complete behavioural analysis. So a second kind of restriction of the analysis capabilities is due to the low level of abstraction in our model.

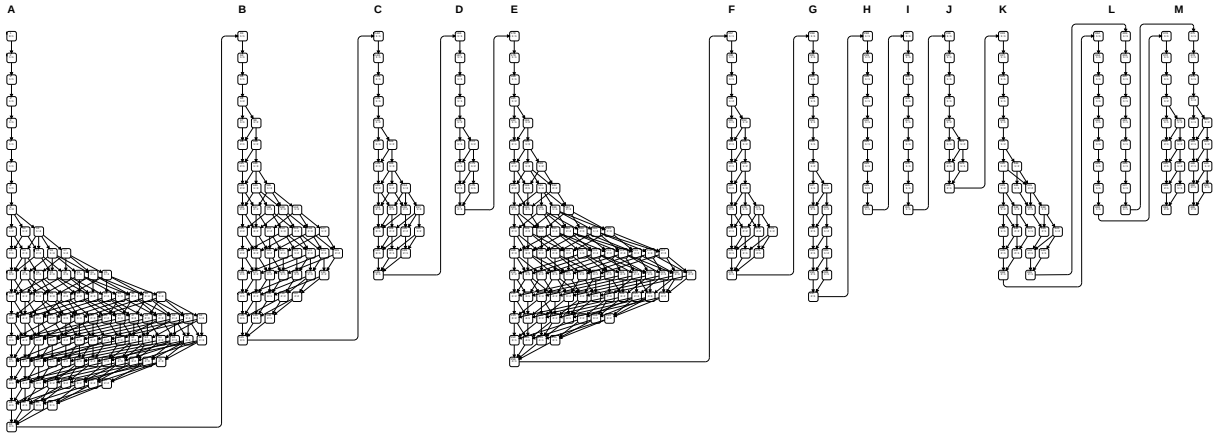

Fig. 10: Example of an occurrence graph

However, state space analysis can be applied to an isolated model given with a specific sequence of stimuli. The occurrence graph of the onboard model depicted in Fig. 10 has been calculated with Design/CPN for a scenario. The nodes of the graph have been aligned in a way that for the first step in processing each stimulus the corresponding node is at the top of the figure and all successor nodes are depicted below unless they do not belong to the possible runs of direct processing of that stimulus anymore. The encircled letters at the top of Fig. 10 refer to the stimuli.

Generally, the onboard system as well as any other component of the ETCS should behave in a determined way. That is, non-determinism is allowed, but with the same input the system is required to generate the same output. Here, output means both external messages and internal values of state variables. Accordingly, a component model is always expected to stop at an unambiguously defined state after having processed the stimulus that has been provided at the last stable state. To proof this behaviour for an isolated component model, all outgoing messages are collected on those places representing external message channels.

Between each two successive stable states there may exist different occurrence sequences. This is due to another principle within our methodology, i. e., to avoid serialisation of functions wherever possible. Assuming that an incoming telegram contains several data packages and that those packages can be processed each by a different function (or the currently active scenario), then the following will happen: Within the pre-processing net the telegram is duplicated as many times as actually needed for different functions and scenarios and each copy of the telegram is directed to one of those functions (or the scenario, respectively). From this point, concurrent execution starts and comes to an end at the next stable state. This mechanism produces a characteristic pattern in the occurrence graph as becomes apparent in Fig. 10.

With the above scheme for interpretation of an occurrence graph with regard to our methodology we are now able to answer the question how we can benefit from a state space analysis of an isolated component model given with a sequence of stimuli. Regarding the occurrence graph in Fig. 10 in particular for the processing of stimulus K, there is no uniquely determined stable state in the following part of the graph. Instead, we discover that the graph evolves in two similar parts again showing the before mentioned characteristic patterns for concurrent processing, but now duplicated, once for each part. From the patterns and the

matching of the stimuli with the stable state, we can infer that there is an error with the processing of stimulus K. We can also guess that the erroneous behaviour occurs in the last marking being common to all evolving occurrence sequences, i. e. the 6th node from the top (below the letter K). Design/CPN allows switching from the occurrence graph analyser to the corresponding state in the simulator, thus enabling to inspect the situation and to identify the error. In the given case, i. e. the train having overpassed the exit point of the RBC (stimulus K), the onboard system has to switch to unfitted mode (level 0) first and then send the exit position report. The corresponding (erroneous) scenario net clearly shows that there is no check of the level attribute of the onboard. Thus, an actual requirement for serialisation of both activities has been identified. This requirement was not explicitly stated within the specification documents [1,2].

As a result an ambiguity has been discovered within the specification of the ETCS. The error could be fixed for the CPN model. In general the state space method can be easily used for validation of open component models according to given sequences of stimuli.

5.2 Model performance

ERTMS/ETCS system modelling has to date proceeded to the following degree of complexity:

	Onboard	**Trackside (RBC)**	**Environment**
Number of nets	75	87	6
Net elements	1,075	1,411	97
Places	734	954	65
Transitions	341	457	32
Hierarchy levels	7	7	3
Line code (incl. comments)	15,500	12,500	0

Table 1: Model complexity

As modelling aims at providing a detailed visualisation of the air gap, the environment model only serves to produce stimulations, which is why the environment model was not taken beyond the required abstraction level.

The model performance can be subdivided into two groups. On the one hand, modelling itself has achieved a number of aims: the SRS has been modelled in a formal way, the interface that has a central role to play for interoperability, i.e. the air gap between track and train, has been visualised, and the operational processes have been shown in the form of scenarios. A second group is represented by the simulation, which implies both simulation of the operational processes and simulation of the supervision functionality.

6 Perspectives

6.1 Additional work on the ERTMS/ETCS model, model extension and refinement

Starting from the present model, additional functional refinements can be made. There is the possibility of examining the effects of various algorithms or implementations of algorithms.

Provisions have already been made in the onboard system for a linking-up with the man-machine interface, and the structure of the interface has already been defined. The basis has thus been laid for connection with an MMI.

Similarly, a means of visualising train movements can be connected at the RBC end. This visualisation is based on a route map, which is supplemented with the dynamic data the RBC supplies for the different trains. This form of visualisation is a reproduction of how the interlocking operator views the system.

6.2 Quality assurance for the ERTMS/ETCS system

Safety standards / Certification support
The formal model can assist in furnishing proof of safety standards compliance. Formal description of the system at the same time provides a clear and unequivocal definition of the system behaviour. On this basis system implementation can be tested with reference to the model, and the system behaviour can be checked for given conditions and also in an abstract manner, in the form of an analytical procedure.

Formal analysis
Reachability analyses have already been made, in which individual scenarios and sequences of scenarios were used as examples. Extended analyses will have to be the subject matter of future work. It has already been indicated that analysing properly has to be accompanied by an interpretation of the findings, if the modelled system is to supply meaningful information.

6.3 Test applications

Transition to real-time simulations
Presently, simulations are made off-line, the time being visualised only in the form of discrete steps of sequences of events. Real-time simulations of models presuppose that in modelling the system, aspects of the real-time mode are given due consideration. What is required in addition is a real-time compatible tool.

Supporting hardware-in-the-loop tests
Adequate modelling and hardware provided, the simulatable model can support tests in the form of hardware-in-the-loop tests. For such tests, part of the real system is replaced by a simulation using suitable hardware, and one or several parts of the system are linked in the form of their implementation. This procedure allows hardware components to be tested individually.

Interface generators
A simulatable model can be used to supply the interfaces of a real or simulated system with stimulations [11].

6.4 Extending the system design with formal means of description modelling, simulation and code generation using the example of SatZB

Modelling work currently under way at the Institute of Control and Automation Engineering uses Petri nets for the specification of satellite-based train control (SatZB) [12]. It is intended to examine with the aid of this model in what way program codes for standard hardware can be derived from a simulatable net. For this purpose, the spiral model (Fig. 3) can be supplemented as follows:

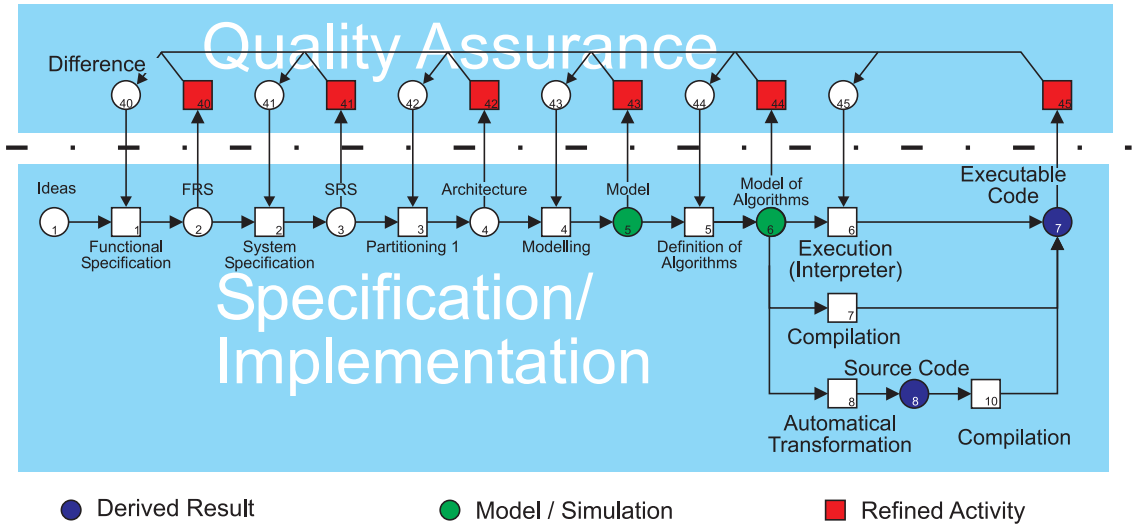

Fig.11: Detail of the spiral model with possibilities of code generation

The following three options of generating executable codes are analysed:

1. Automatic source code generation, which is followed as a second step by conventional compilation.
2. Compilation of the Petri net itself to produce an executable code.
3. Execution of the net itself in the form of a system. Within the meaning of the definition of the term 'model' this implies that the system is substituted by a model and its hardware.

7 Summary

On the basis of the System Requirements Specification and the relevant documents [1,2] an ERTMS/ETCS model was developed. This model visualises the system in the form of three Petri net models. The onboard and trackside systems together form the core which is embedded in a model of the environment. The two core models reflect the aspects of the sub-system context, of the processes proceeding within the sub-systems, of operational processes in the form of scenarios and specific functions. In developing these models, a combination of three elements was used: Petri nets as a means of description, an integrated method and the tool Design/CPN. It was demonstrated that during the phases of system development, covering the system specification through to the final system design, a model based on Petri nets can be used.

Work can continue, on the one hand, with extending and refining the existing ERTMS/ETCS model, while on the other hand the next phases of system design, including the executable code, are to be analysed on the basis of satellite-assisted train control.

Acknowledgement

The results described in this paper are the experiences a project with the Deutsche Bahn AG. We thank our partners especially Mr. B. Ptok.

References

[1] EEIG ERTMS Users Group: System Requirements Specification - SRS. 96E236, Version 4.0, Brussels, 1997.

[2] EEIG ERTMS Users Group: Scenarios. 96E283, Version 2.0, Brussels, 1997.

[3] EEIG ERTMS Users Group: TIU FFFIS. 97E117 Version 1.0, Brussels, 1997.

[4] H.-J. Schneider: Lexikon Informatik und Datenverarbeitung. 4. Edition, R. Oldenbourg, Munich, Vienna, 1997.

[5] GMA Unterausschuß 1.8.1 Standardisierte Beschreibungsmittel in der Automatisierungstechnik: Glossar. Brunswig, 1998. www.ifra.ing.tu-bs.de/gma181/glossar.htm.

[6] K. Jensen: Coloured Petri Nets, Volume 1, Monographs in Theoretical Computer Science. Springer, Berlin u.a. , 1992.

[7] Design/CPN: Occurrence Graph Analyser-Manual. Version 3.0, Aarhus, 1996.

[8] E. Yourdon: Modern structured Analysis. Prentice Hall, 1989.

[9] B. Boehm: Software Engineering Economics. Prentice Hall, 1981.

[10] P. Starke: Analyse von Petrinetzen. Teubner, Stuttgart, 1990.

[11] H.-M. Schulz, M. Meyer zu Hörste, B. Ptok, E. Schnieder: Modelling and simulation of the European Train Control System for test case generation. In: B. Mellit, R. J. Hill, J. Allan, G. Scuitto, C. A. Brebbia: Computers in Railways VI (COMPRAIL), Lisbon 1998, pp 649 – 658, WITPress.

[12] F. Reißner: Satellitengestützter Zugleitbetrieb - betriebliches Lastenheft, Edition 1.0, Munich, 1998.