Formal Specification and Development of a Safety-Critical Train Management System

A. Chiappini[1], A. Cimatti[2], C. Porzia[1], G. Rotondo[1], R. Sebastiani[2], P. Traverso[2], and A. Villafiorita[2]

[1] Ansaldo Segnalamento Ferroviario, Via dei Pescatori, 16100 Genova, Italy
[2] IRST, Istituto per la Ricerca Scientifica e Tecnologica, Povo, 38050, Trento, Italy

Abstract. In this paper we describe the on-going specification and development of Ansaldo's Radio Block Center (RBC), a component of the next-generation European Rail Traffic Management System (ERTMS). The RBC will be responsible of managing the movement of trains equipped with radio communication. Its development process is critical: the RBC is a large-scale and complex system, it must provide several novel services at different levels of functionality, it must guarantee interoperability according to European standards, and, last but not least, a high level of safety. We have addressed these issues by devising a development based on formal specifications. ERTMS scenarios have been formalized in order to provide a better understanding of the interoperability requirements. The architecture of the RBC has been formally specified such that the system can be incrementally built as an overlay system (compatible with the existing train detection and interlocking systems) and modularly expanded to control different kinds of trains. The formal specifications of the behaviour of each RBC module have been structured hierarchically: they provide an easy-to-understand documentation for customers and developers; moreover, they can be simulated and validated automatically at the early stage of the development process, thus providing a high level of confidence in their safety in a cost-effective way.

1 Introduction

The next-generation European railway control systems will have to address a set of novel safety, performance, interoperability, and compatibility requirements. One of the goals is to build a safety critical system which improves railway system performances while reducing investment and operating costs. Another goal is to build systems interoperable among different European nations: in practice, a train working for a given railway (e.g. the Italian one) should be able to work in neighboring countries (e.g. in the Austrian, French and German railway's networks). Moreover, the system should be compatible with the existing ones. Finally, it should provide new functionalities incrementally, allowing thus for different levels of increasing sophistication.

The achievement of all these novel and different requirements is a crucial issue for the future of the European railway system. This has led to the set up of a project at the European (rather than national) level, called the European Rail Traffic Management System (ERTMS). In the ERTMS project, the performance requirement has been addressed by conceiving a completely new communication system between trains and signalling systems on the ground, based on GSM mobile radio data transmission and on high-rate balises (EuroBalises). The interoperability requirement has been addressed by designing a European standard communication protocol between the train and the ground system, that has been described by a set of uses cases, called ERTMS scenarios. The system is described in [3] and [2].

The goal of the project described in this paper is the Ansaldo's specification and development of the Radio Block Center (RBC), the ERTMS component responsible for the management of the movement of trains equipped with radio communication. The RBC is a large-scale and complex system: it must interact with several systems beyond trains, e.g. with existing interlocking systems, with the RBC's controlling neighboring parts of the railways line, with a man-machine interface for monitoring and control. It must provide several novel services and guarantee interoperability according to the ERTMS standards, and, last but not least, a high level of safety. It is clear that its specification and development are critical.

We have addressed these issues by devising a development based on formal specifications. One of the main motivations for this choice is the fact that an approach based on formal methods allows for unambiguous specifications which can be validated at the early stage of the development process. Unambiguous specifications are extremely important for the interoperability and compatibility requirements. Their "early validation" is a must in practice for the safety requirements and for the development of a large-scale system. One of the main goals of the project is to devise a development process based on formal techniques which can be cost-effective and really usable by the experts in the signalling field. This has been achieved through some main steps where we have managed to adopt formal notations and formal verification techniques which can be smoothly introduced into the Company's development process. An overview of these main steps is presented in Section 2. The rest of the paper (after a brief description of the application domain in Section 3) describes in detail these steps: the formalization of ERTMS scenarios (Section 4), the formal input/output specification of RBC's software modules (Section 5), the hierarchical formal specification of their behaviour (Section 6) and their early validation (Section 7). In order to keep the paper short and readable, these steps are described through a simple example, which is discussed all through the paper. The whole formal specification and design at the current stage of the project is presented in a detailed project report [1].

2 Overview of the Approach

Cost-effectiveness and real usability of formal methods within the project has been achieved through a specification process based on the following main steps.

First, ERTMS scenarios describing the RBC-train communication have been formalized in order to provide a better understanding of the interoperability requirements. At this stage, it is important to use a language which is easy to understand by signalling experts. Scenarios have been formalized in the "Message Sequence Chart" (MSC) format [6]. MSC's have been designed to represent the dynamic behaviour of protocols. In spite of the fact they are formal and thus provide unambiguous descriptions, their graphical notation is extremely intuitive and easy to read. MSC's have allowed experts in signalling and experts in formal methods to share their respective know-how. As a result, they have allowed us to disambiguate several scenarios and to state clearly the hypotheses under which the scenarios are applicable.

Second, we have designed the high level architecture of the RBC through "Structured Description Language (SDL) block (interconnection) diagrams" [5]. SDL diagrams allow for the definition of software modules and the information exchanged between modules. We have first defined the interconnection diagram between the RBC as a black box and the other systems it interacts with (e.g. the interlocking, the other RBC's, the interface to the operator and the train). We have then decomposed the RBC module into more "elementary" functionalities.

Third, starting from the formalized scenarios (MSC's) and the architectural decomposition in different software modules (SDL block diagrams) we have formally specified the dynamic behaviour of each module. We have used "State Charts" [4] for this purpose. State Charts are a well-known formalism which is used to describe systems components as automata. Automata are a well known technique used in Ansaldo—engineers can easily understand them and use them to describe the system behaviour. The main problem in this task is due to the fact that the system is quite large. A mere one-level and flatten description of the system in terms of state charts would be unpractical and very difficult to maintain. For this reason, state charts have been structured hierarchically: we have developed state charts which describe the behaviour of the system at a high level, abstracting away several details. These state charts are simple (i.e. with a reasonable number of states) and highly non-deterministic: given a state of the system, the system can evolve into more than one next state. In other words, the state chart is not detailed enough to specify the exact conditions under which the system evolves in one of the possibly many behaviours. These automata provide a high level view of the system, are extremely important for the documentation (both for developers and customers), and can be easily validated by railways experts by means of simple inspections.

Then high level state charts are refined in detailed ones. We have specified the detailed behaviour of the system through "SDL processes". SDL processes describe in detail how the system evolves from one state to the next. SDL [5] is a language which has been extensively used in industry. Its notation is not difficult for an engineer which is used to traditional programming and software engineering notations, like data flow diagrams and flowcharts. The signalling expert can still read the detailed specification easily, but a manual validation by inspection is unpractical, since at this level of detail the specification is large.

As a consequence, we have exploited existing tools to automatically simulate the specifications. This is the immediate advantage of having developed formal specifications: they can be simulated far before the code implementation phase. Moreover, we have used "Model Checking" techniques to explore all the possible behaviours of the system exhaustively. The significant added value of Model Checking w.r.t. simulation is that it can guarantee that a safety property is satisfied for all possible behaviours of the system. Indeed, Model Checking has revealed unexpected behaviours of the system as initially specified: specifications errors have been corrected until the specifications have been validated by model checking.

In this project, we have used the ObjectGeode tool [7], which among other languages and features, allows for the use of MSC's, SDL bock diagrams, State Charts, SDL processes, and a simulator and a model checker for SDL processes.

3 Overview of the Application

The main functionality of the RBC is the allocation of Movement Authority (MA) to trains, that is, permission to move for a certain distance. In order to safely allocate MA, the RBC collects information about (but not limited to) the position of trains on the line, emergency situations, and pending orders from human operator(s).

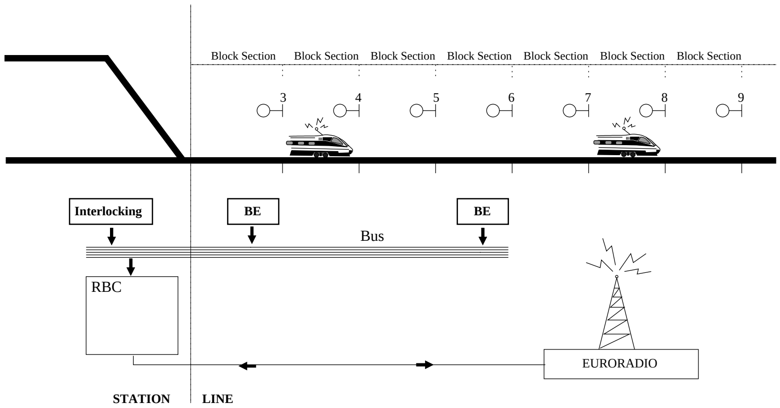

Fig. 1. The ERTMS System.

Figure 1 depicts the situation. The line is logically divided into block sections; the signalling system must guarantee that any block section is occupied at the most by one train. In order to do so, the entrance into a block section is protected by a signal. A train detection system senses the status of occupation of the blocks and sets the signals accordingly. In the station the situation is more complex: an interlocking system is responsible of setting/releasing itineraries, by safely controlling switches and signals. The RBC is connected to all the interlockings installed in the zone it controls. The RBC collects information about the position of trains, the status of occupation of blocks on the line and the itineraries set in the stations, computes MA's based on the information it has collected, and sends them to the trains.

4 Scenarios (via Message Sequence Charts)

The least restrictive way of guaranteeing interoperability is to impose a standardization of the messages and the protocol between RBC and train. The ERTMS project has thus defined a list of correct RBC-train messages, called telegrams, and a set of use cases, called scenarios, that provide examples of the communication protocol. Scenarios are composed of a description in natural language, that explains the rationale of the scenario (e.g. the applicability) and a tabular description of the protocol, that provides examples of how telegrams are exchanged.

Scenarios leave several options open: we must choose among all the options provided by the scenarios the ones that are most suitable, e.g., with the system in which the RBC is going to operate. For instance, the Scenario describing MA allocation to trains does not fix the criteria according to which the RBC decides to allocate MA's: such criteria indeed depend upon the rules governing the signalling system in which the RBC operates (e.g. they are decided on a national basis). This information, however, is essential to provide an executable specification of the RBC.

More to the point, the tabular form used in [2] is kind of ambiguous. The succession of rows does not necessarily correspond to progression of time. Alternatives are difficult to represent.

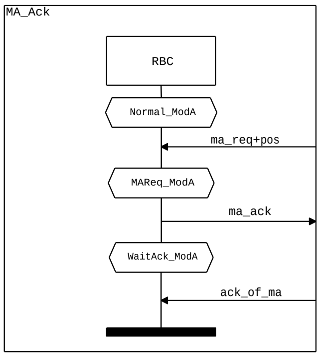

Fig. 2. MSC of MA Allocation with acknowledgement.

The above considerations suggest the need for a more precise notation, with clear syntax and semantics, and easily understandable by experts of the railway system. We adopted the Message Sequence Charts (MSC) notation, as it meets all these requirements. Figure 2 shows an MSC that rationalizes part of the Scenario describing MA allocation to trains. We distinguish an entity, called `RBC`, that communicates with the environment (framing rectangle). The environment comprehends everything external to the `RBC`: in our case, trains. The `RBC` and the environment exchange messages, represented in the MSC by labelled arrows. Time progresses from the top to the bottom. The MSC represents the reception of a request for a MA (`ma_req+pos`), to which the RBC answers with a MA requiring an acknowledgement (`ma_ack`). Upon receiving the MA, the train acknowledges the reception (`ack_of_ma`). The MSC is also marked with "conditions" (e.g. `Normal_ModA`), that fix the states of the communication between track and train.

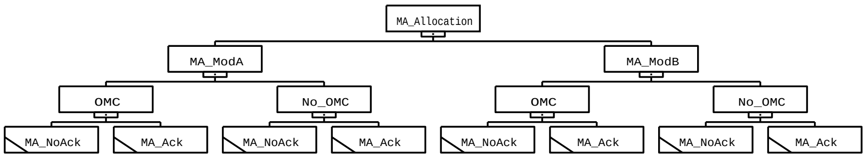

Fig. 3. Abstract MSC of MA Allocation.

In ObjectGeode, MSC's can be combined to form more complex MSC's, using operators like sequence, alternative, and iteration. The composition of MSC's is represented by a tree (called hierarchy), in which the nodes are operators and the leaves are MSC's. Figure 3 shows a hierarchy of MSC's that rationalizes the behaviours of the Scenario describing MA allocation to trains. We individuate two different operating modes, called Mode A and Mode B. In Mode A (leftmost subtree), trains always ask for MA's, whereas in Mode B (rightmost subtree) the RBC allocates MA to trains when it decides to do so. The RBC may request a a change in the operating mode (subtrees marked `OMC`—Operating Mode Change) and, finally, MA's may require an acknowledgement or not (leaves of the tree). The MSC in Figure 2 is the leaf `MA_Ack` in the hierarchy in Figure 3.

5 Architectural Description (via SDL Interconnection Diagrams)

MSC's help understand and formally define the way in which the RBC interacts with trains. This is not the whole story, however, as the RBC interacts with several other actors. The second step in the specification of the RBC, thus, is the definition of the environment in which the RBC operates (what we call the "context diagram") and the definition of the constituent blocks of the RBC (architectural decomposition).

Figure 4 shows the context diagram for the RBC. The RBC interacts with actors, that range from devices for the configuration of the RBC at startup

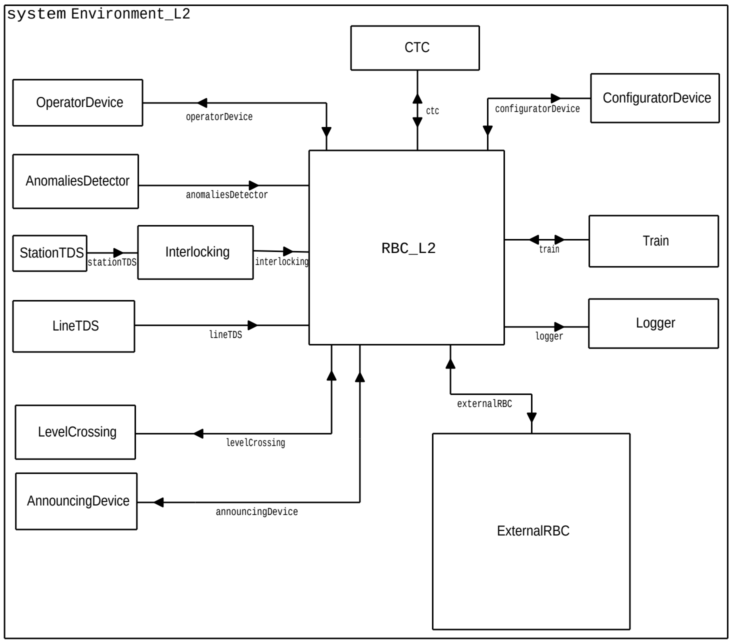

Fig. 4. Context Diagram of the RBC.

(Configurator Device), through devices for collecting the status of the line (Operator, Anomalies Detector, Interlocking, Line TDS, External RBC), to trackside devices directly operated by the RBC (Level Crossing and Announcing Device).

6 Dynamic Behaviour Specification (via State Charts and SDL processes)

We can now start filling the blocks of the interconnection diagram, using the MSC's as a constraint for the I/O behaviour of the RBC. To simplify the process of specification of the blocks, and increase our confidence on the specifications, we start with abstract specifications—using State Charts—that we later refine into SDL code. The abstract specifications hide several details of the behaviour of the RBC (like, for instance, what decisions trigger a transition to a state rather than another): the goal here is providing "high level" specifications that can be validated "by hand".

Figure 5 shows the abstract specification of the communication protocol described in the Scenario describing MA allocation to trains. The left hand side implements the allocation of MA in Mode A. We distinguish three states: `Normal_ModA`, `MAReq_ModA`, and `WaitAck_ModA`. In `Normal_ModA` the RBC waits for a MA request from the train (`ma_req`), that causes a transition into `MAReq_ModA`. From `MAReq_ModA` the RBC sends a MA to the train and non-deterministically

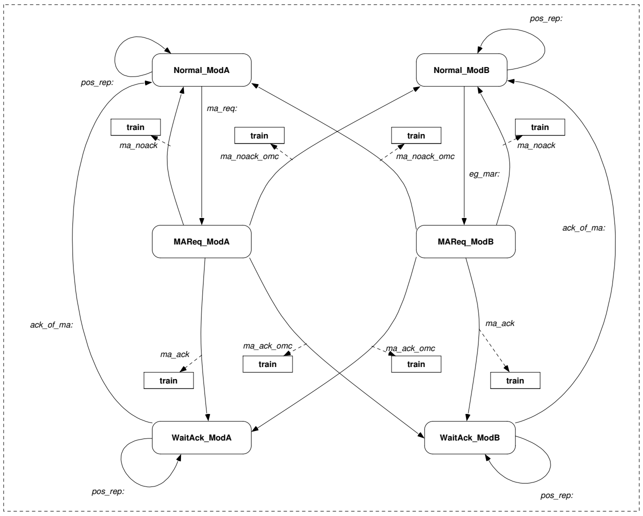

Fig. 5. Abstract State Chart for MA Allocation.

goes into one of the following four states, depending on the MA the RBC decides to send: `Normal_ModA` (MA that does not require acknowledgement nor operating mode change), `WaitAck_ModA` (MA that requires acknowledgement but does not require to change operating mode), `Normal_ModB` (MA that does not require acknowledgement, but that requires an operating mode change), `WaitAck_ModB` (MA that requires both acknowledgement and to change operating mode). The state chart we just presented does not describe the internal mechanisms and strategies according to which the RBC chooses what MA to send, and, therefore, its next state: our goal, here, is the specification of the generic behaviour of the RBC (e.g., the fact that from `MAReq_ModA`, the RBC has four possible next states, corresponding to four logically different states of the communication).

State Charts can be automatically translated by ObjectGeode into SDL code. In our case, (a slightly modified version of) the chart of Figure 5 has been automatically translated into SDL code, that has later been refined by hand, adding the details to make its behaviour deterministic. Figure 6 shows the portion starting from `MAReq_ModA`: the RBC gets information about the occupation of the line (`track_path`), computes the MA (`cma`), then decides the next state. Now the transition onto the next state is deterministic and depends upon a call to the function `omc` (that determines whether to change operating mode) and a parameter `cfg_ack` (that determines whether to ask for acknowledgements). Depending on which of the four branches is chosen, the RBC sends a different message to the train (one of `ma_ack_omc`, `noma_ack_omc`, `ma_ack`, and `ma_noack`).

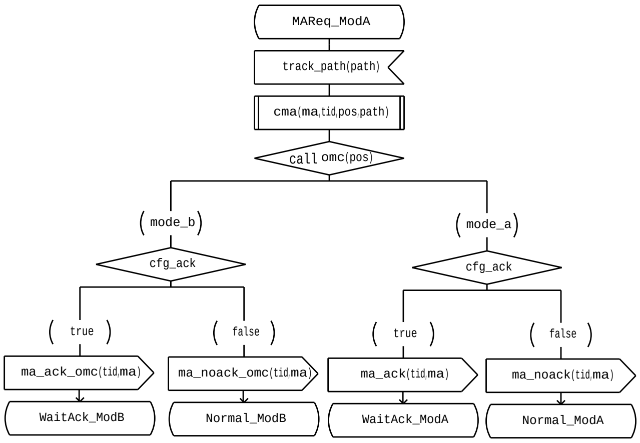

Fig. 6. SDL representation of the State Chart.

7 Early Validation (via Simulation and Model Checking)

The last step is the validation of the specifications. In this step we exploit the executability of the specifications, by running them inside ObjectGeode's simulator and model checker (exhaustive simulator). The goal of this activity is verifying that certain properties hold of the specifications of the RBC. It is important here to observe that, short of a few properties that can be proven with the RBC in isolation (like the absence of deadlocks), in order to verify whether the RBC behaves correctly, we must immerse the RBC into an environment. The environment is composed of, for instance, interlocking and the train detection systems, that provide inputs to the RBC, and trains, that communicate with the RBC and move according to the MA's they receive from the RBC. Thus, by monitoring how trains move, we can check whether the RBC behaves as it should. We can arbitrarily degrade the performances of the environment, by allowing, for instance, telegrams to be lost or corrupted by the radio channel. This allows to test the behaviour of the RBC in a wide range of (critical) situations.

Properties are verified by running simulations (either random or exhaustive) with watches that monitors the behaviour of the system. Such watches can either express properties holding of a state or properties holding along a sequence of states. For instance, we can verify that no block section is occupied by more than one train or that a certain train has reached a certain position.

8 Conclusions

In this paper we have described an on-going project whose aim is the formal specification of a system (the implementation of RBC done by Ansaldo) for the management of trains according to the ERTMS requirements. We have adopted formal methods in order to cope with the safety and interoperability requirements. One of the main issues of the project is the provision of formal specification techniques which can be easily understood and used cost-effectively within the Company's development process. This issue has been addressed by providing formal specifications which are unambiguous and, at the same time, can be easily inspected by the development team; by devising an hierarchical specification of the system which provides an highly structured documentation for customers and developers; by an early validation phase which is performed during the specification and design phase and in which (detailed) formal specifications can be simulated (either interactively or exhaustively) to detect possible expensive specification errors at the early stage of the project.

We have approached the problem of the specification of a quite large and complex system, interacting with several external actors, in a selective and incremental way. At this stage of the project we have formally specified and validated a significant subset of the functionalities. Some of them are (part of) the management of trains inside the area controlled by the RBC, the entry/exit of a train from/to areas controlled/not-controlled by other RBC's, some cases of anomalies handling. These specifications and the methodology used to provide them are now taken as examples and guidelines to complete the specifications and design in the current continuation of the project.

Acknowledgements

The work described in this paper was founded under contract 1607/250000(NA). The authors would like to thank all the people at Ansaldo and IRST involved in the project.

References

1. A. Chiappini, A. Cimatti, F. Giunchiglia, G. Rotondo, R. Sebastiani, P. Traverso, and A. Villafiorita. Formal Specification of the Radio Block Centre (RBC): First Part. RBC I: Ansaldo-IRST Project Report, December 1998.
2. EEIG-ERTMS Users Group, 25 Avenue De Beaulieu 1160 Brussels, Belgium. *Scenarios.*
3. EEIG-ERTMS Users Group, 25 Avenue De Beaulieu 1160 Brussels, Belgium. *System Requirements Specification.*
4. David Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8(3):231–274, June 1987.
5. ITU-T. *CCITT specification and description language (SDL)*, March 1993. ITU-T Recommendation Z.100.
6. ITU-T. *Message Sequence Chart (MSC)*, October 1996. ITU-T Recommendation Z.120.
7. VERILOG. *ObjectGeode Documentation.* Available at *www.verilogusa.com.*