









Designing a Demonstrator of Formal Methods for Railways Infrastructure Managers

Davide Basile¹, Maurice H. ter Beek¹, Alessandro Fantechi^{1,2},
Alessio Ferrari¹, Stefania Gnesi¹, Laura Masullo³, Franco Mazzanti¹,
Andrea Piattino³, and Daniele Trentini³

¹ ISTI-CNR, Pisa, Italy

{d.basile,m.terbeek,a.ferrari,s.gnesi,f.mazzanti}@isti.cnr.it

² Università di Firenze, Firenze, Italy

alessandro.fantechi@unifi.it

³ SIRT I S.p.A., Genova, Italy

{l.masullo,a.piattino,d.trentini}@sirti.it

Abstract. The Shift2Rail Innovation Programme (IP) is focussing on innovative technologies to enhance the overall railway market segments. Formal methods and standard interfaces have been identified as two key concepts to reduce time-to-market and costs, while ensuring safety, interoperability and standardisation. However, the decision to start using formal methods is still deemed too risky. Demonstrating technical and commercial benefits of both formal methods and standard interfaces is necessary to address the obstacles of learning curve and lack of clear cost/benefit analysis that are hindering their adoption, and this is the goal of the 4SECURail project, recently funded by the Shift2Rail IP. In this paper, we provide the reasoning and the rationale for designing the formal methods demonstrator for the 4SECURail project. The design concerns two important issues that have been analysed: (i) the usefulness of formal methods from the point of view of the infrastructure managers, (ii) the adoption of a semi-formal SysML notation within our formal methods demonstrator process.

1 Introduction

The European public-private Joint Undertaking (JU) for rail research Shift2Rail (S2R)¹, acting under the broad umbrella of the EU Research and Innovation programme H2020, aims to improve the state-of-the-art of rail technology and revolutionise rail as a mode of transport making it a backbone of future mobility.

The Shift2Rail Innovation Programme 2 is focussing on innovative technologies with a view to enhance the overall railway market segments. Two key concepts that have been identified to reduce the time for developing and delivering railway signalling systems are formal methods and standard interfaces. They are

¹ <http://www.shift2rail.org>.

also useful to reduce high costs for procurement, development, and maintenance. Standard interfaces are needed to increase market competition and standardisation, and to reduce long-term life cycle costs, whereas formal methods are needed to ensure correct behaviour, interoperability, and safety. The Shift2Rail initiative plans to demonstrate technical and commercial benefits of formal methods and standard interfaces, applied on selected applications, with the goal of widening the industrial uptake of these key aspects. However, the decision to start using formal methods is often deemed too risky by management, and the railway sector is notoriously cautious about the adoption of technological innovations compared with other transport sectors [3]. Thus, demonstrating technical and commercial benefits of formal methods and standard interfaces is necessary to address the obstacles of learning curve and lack of clear cost/benefit analysis.

This paper presents the first results of the Shift2Rail project 4SECURail², and in particular of the workstream 1 named: “Demonstrator development for the use of Formal Methods in Railway Environment”. This workstream aims to provide a *demonstrator* of state-of-the-art formal methods and tools, applied on a railway signalling subsystem described by means of standard interfaces. This demonstrator will be used to evaluate the learning curve and to perform a cost/benefit analysis of the adoption of formal methods in the railway industry. In this paper, we discuss the overall process to follow for the rigorous construction of system specifications (the formal methods demonstrator process), together with the suitability criteria for the supporting tools and the description of the architecture of the demonstrator itself. The main contributions are:

- a description of the planned architecture of the demonstrator, e.g., the expected types of (semi-)formal models to develop during the process;
- a discussion on the role of UML/SysML [42, 44] as standardised notation within the demonstrator and the role of the internally generated (semi-)formal models with respect to the final system requirements specification that the demonstrator process is expected to define;
- an identification of the kind of data about the cost of the approach that needs to be assessed, having as target the cost-benefit analysis.

The paper is organised as follows. In Sect. 2, the 4SECURail project is recalled. Sections 3, 4 and 5 describe the point of view of the infrastructure managers, the architecture of the demonstrator, and the adoption of standard notations. Finally, Sect. 6 discusses conclusions and future work.

2 The 4SECURail Project

Despite several success stories on railway applications of formal methods (cf., e.g., [5, 15, 22–24, 27]) these mathematically-based methods and tools for system development still find significant *obstacles* to their adoption in railway software industry.

² <https://www.4securail.eu/>.

Obstacles to the Adoption of Formal Methods. The major obstacle is the high learning curve; formal methods have the image of being too difficult to apply for ordinary engineers [3, 5, 29]. Other significant obstacles include the fact that applicable standards, such as CENELEC EN 50128 [21] do recommend formal methods, but do not provide sufficiently clear guidelines on how to use them in a cost-effective way and there is no clear picture of what can be achieved using formal methods (in terms of benefits, both technical and economical). This leads to the transition to formal methods being deemed too risky by the management.

Another obstacle to the widespread use of formal methods is the lack of commercial tools, easily integrated in the software development process and working on open standard formats [29]. In fact, the current state of the art of the development tools market either offers industry-ready, well maintained and supported tools working on closed proprietary formats, or open source tools working on standard open formats, but offering low levels of support and maintenance.

Formal Methods Demonstrator, Standard Interfaces and Cost-Benefit Analysis. To address these obstacles, 4SECURail foresees the development of a demonstrator of state-of-the-art formal methods to be used as a benchmark to demonstrate technical and commercial benefits of formal methods application on a selected application (a railway subsystem). The formal development demonstrator prototype will consist of the detailed description of the process that will be followed to provide a rigorously verified model of the application under development, together with the list of the tools to be employed. The demonstrator will also take into due account the adoption of *standard interfaces* among the components of the selected applications. The role of standard interfaces have also been investigated in a previous Shift2Rail project, named X2Rail-2³.

Interested Railway Stakeholders. In the railway domain, it is expected that the following stakeholders will be interested in the use of formal methods.

- Systems designers: formal methods will contribute to the early validation of the consistency of captured requirements and the check of compliance of design solutions with user and safety requirements.
- System and product developers: formal methods will provide an environment for developing the project and the possibility of using simulators for testing.
- Infrastructure managers: main railway networks are under responsibility of independent *Infrastructure Managers* whose interest is in increasing the interoperability among different equipment suppliers, improving their competitiveness and maximizing safety and reliability, at the same time reducing life-cycle cost of signalling system: all goals supported by the adoption of formal methods and standard interfaces.

4SECURail will address the views of these stakeholders, with special focus on the infrastructure managers, exactly because it is expected that they will benefit most, both in terms of safety and of cost, from the adoption of formal methods and standard interfaces.

³ https://projects.shift2rail.org/s2r_ip2.n.aspx?p=X2RAIL-2.

4SECUrail Roadmap. 4SECUrail will be based on the results and indications provided by the X2Rail-2 project to select suitable tools for supporting the development of a formal model and its verification. Moreover, tools and languages for the description of standard interfaces (e.g., Standard UML, SysML, etc.) of the selected railway subsystem will also be considered and integrated in the demonstrator framework. The exercising of the demonstrator on an identified railway subsystem will address the current obstacles and the lack of a clear cost/benefit analysis for the appraisal of the application of formal methods. This implies the assessment of different learning curves, the collection of relevant data about cost and benefits, and provide results about the financial feasibility and economic sustainability of the adoption of formal methods at micro and EU level. The above goal will be achieved by implementing the following activities, implemented in a specifically deployed Work Package (WP2) of 4SECUrail.

- Development of the **formal development demonstrator prototype**, that will consist in the identification of the overall process to be followed, phase after phase, for the formal development, and establishing the criteria for suitability of tools supporting each phase. In particular, the formal development demonstrator will be based on the use case developed in X2Rail-2 [38, Sect. 5.4.1]. The definition of the architecture of the demonstrator will include the choice and integration of appropriate formal methods and tools, taking into account the results produced by the Shift2Rail projects ASTRail and X2Rail-2. Moreover, in this activity we will identify the tools for the description in standard interfaces of the railway subsystem.
- Selection of the railway signalling subsystem and its use for the **exercising of the formal development demonstrator prototype**.
- **Cost-benefit analysis**, and identification of learning curve scenarios (and related cost) connected to the adoption of formal methods. The cost-benefit analysis will identify the financial feasibility and the economic impact of the implementation of formal methods against the baseline scenario, made by processes which do not exploit formal methods.

3 The Point of View of the Infrastructure Manager

An Infrastructure Manager (IM) has to provide a validated specification of a desired equipment to the Manufacturers. In a classical client/developer scenario (see Fig. 1 left) the common practice is the generation of—usually informal—system requirements document. This document can then be used by the developer to build an initial executable specification of the system, and then refine it (possibly using formal or correct-by-construction methods) into a final product.

The scenario in case of railway IMs is slightly different, since the main interest is in providing the same rigorous/verifiable specification not just to single developers, but to possibly multiple different developers that should produce equivalent products (see Fig. 1 right). This is precisely the case described by the X2Rail-2 use case (see Sect. 2), where defining a standard/rigorous/verifiable specification of the system to be developed becomes the IM's responsibility.

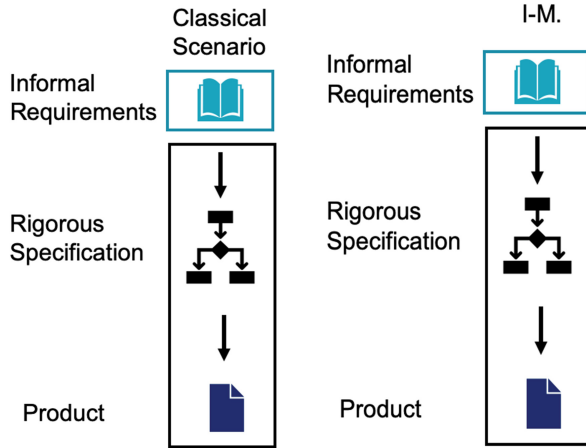


Fig. 1. The classical client/developer scenario (left) and the client/multiple developers scenario (right)

Actually, in the case of railway IMs, the scenario is even more complex. In fact, the railway infrastructure is constituted by a multitude of *subsystems* (each one possibly developed by a different supplier) that must correctly interact among them. In this case, the problem of building rigorous/formal/verifiable specifications should extend also to the verification of the interactions between these components (see Fig. 2). Clearly this does not hold only for railway IMs, but also for any other kind of complex infrastructures (e.g., telecommunications).

This introduces a further dimension of complexity. For example, safety properties can often be verified by reasoning at the level of single subsystems (e.g., ensuring that independently from the possible external interactions, no unsafe conditions are even reached), but the same cannot be said for specific properties related to the composite behaviour of several subsystems (e.g., liveness, absence of deadlocks, or missing desired execution paths involving the behaviour of several subsystems). A special case of these scenarios is when the produced specification takes the role of “standard specification” supported by international organisations (like the International Union of Railways (UIC)⁴, the European Union Agency for Railways (ERA)⁵, or UNISIG⁶), an industrial consortium to develop ERTMS/ETCS technical specifications), defined with the aim of creating interoperable railways in the whole of Europe (Single European Railway Area, SERA). For example, Fig. 2 depicts interoperability between Radio Block Centre (RBC) and Interlocking (IXL) by means of the RBC-IXL standard interface; and between IXL and Level Crossing (LX) by means of the IXL-LX standard interface.

⁴ <https://uic.org/>.

⁵ <https://www.era.europa.eu/>.

⁶ http://www.ertms.net/wp-content/uploads/2014/09/ERTMS_Factsheet_8_UNISIG.pdf.

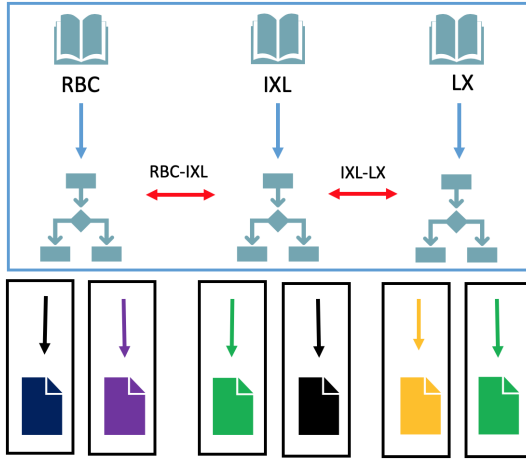


Fig. 2. The client/multiple developers scenario within a complex infrastructure

The Role of Standard(ised) Interfaces. The goal of our demonstrator related to the adoption of “standard interfaces” includes two standardisation aspects, since it aims at exploiting the use of formal methods for the definition of *standardised interfaces* (goal: interoperability) described in *standard notation* (e.g. SysML) (goals: uniformity, understandability, non-ambiguity). A very detailed presentation of the expected benefits from the adoption of standard notations for standardised signalling interfaces can be found in [13].

4 The Overall Structure of the Demonstrator Process

We now describe the overall structure of a demonstrator process aimed at employing formal methods to support IMs, distinguishing three possible cases. The next section will be dedicated to the specific instantiation of this process in the 4SECU-Rail context. The overall structure of a generic software development process targeted to the definition of rigorous system specifications which exploits the use of formal methods (our demonstrator) can be described as in Fig. 3.

First Case (with Requirements Elicitation). Starting from some input describing the initial IM requirements of the system, we start an *agile*⁷ development phase in which the requirements are transformed into “formal simulatable models”. These models are developed incrementally, and continuously analysed by means of formal verification, simulation, animation, and test case generation. These abstract formal models can also be refined by adding additional details into “refined simulatable models” that may help in validating the system behaviour, possibly through simulation and animation. Once these formal models are sufficiently stable, they represent the base for the construction of the demonstrator output (the official system requirements specification), in the form of description of “abstract system requirements”, “safety requirements”, “detailed system

⁷ <https://www.agilealliance.org/>.

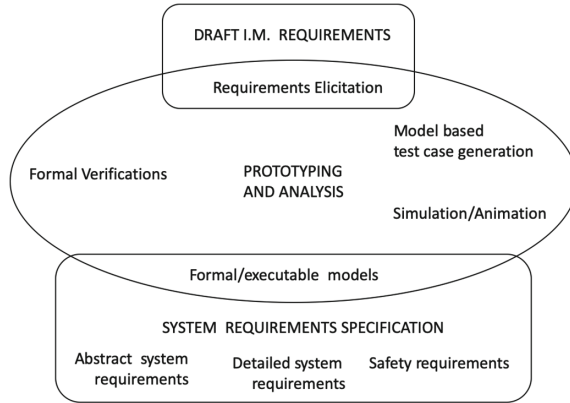


Fig. 3. Overall generic structure of demonstrator (first case)

requirements” (see Fig. 3). The resulting system requirements are still likely to be expressed in natural language, but enriched with tables and diagrams extracted from the (semi-)formal models. The (semi-)formal models themselves might be made available as complementary documentation.

On one hand, the construction of multiple, different semi-formal/simulatable/formally verifiable models allows to obtain a deep understanding of the system design from many points of view and many levels of abstraction. On the other hand, this multiplicity raises the problem of keeping these models somewhat “synchronised”. For example, if, for any reason, one of the models needs to be modified because of the discovery of some defect, the impact of the change on the other models surely cannot be ignored. This may require the construction and maintenance of some kind of cross-references between these artifacts, and probably also between the artifacts of the final “system requirements specification” resulting from the process. The effort needed for keeping all the different artifacts well synchronised should not be underestimated and might play a non-trivial role in deciding how many “points of view” to take into account.

Second Case (without Requirements Elicitation). The whole schema still holds in the case in which the input of the overall demonstrator process is not constituted by *Draft IM Requirements*, but by an already consolidated/official set of system requirements/safety requirements, that should be the object of more rigorous analysis. In this case, we simply would not have the *Requirements Elicitation* activity oriented to the consolidation of the *Draft IM Requirements* (i.e., the upper block in Fig. 3 is not present). In this second case, given that the starting point is an already consolidated specification, the modelling activities (in terms of tools and methods) might be somewhat different from the incremental prototyping activity driven by a rigorous/formal *Requirements Elicitation* phase.

Third Case. The same overall schema might also work in the mixed case in which an already consolidated set of system requirements/safety requirements might

have to be extended/updated by an additional set of new user requirements (somewhat of a composition of the previous two cases). In these cases, the availability of previous formal simulatable artifacts would be of great help for the process. We consider as already acknowledged (cf., e.g., the related Shift2Rail surveys in [3, 5, 26, 38]), that there is not a single formal method or tool that can fit all the possibly desired verification and modelling needs in the railway field [28, 39]. Therefore, the whole *Modelling and Analysis* activity is supported at its best by a rich integrated ecosystem of tools and methodologies, rather than a single monolithic, usually closed, and tied to single specific methodologies framework. We recognise, however, that at least in the first case, where a classical V-shaped process might be followed covering all the steps from *Requirement Elicitation* to *Official Requirements Specification generation and verification*, a reference modelling framework might actually help in building and maintaining all the documentation related to the various artifacts being generated.

The Expected Output of the Demonstrator Process. The set of artifacts in output from the formal methods demonstrator process are represented in our overall generic model by the final “System Requirements Specification”. Actually, these artifacts might be of different nature and with different purposes, we identify four cases:

- A *rigorous natural language textual description*, possibly enriched with standard diagrams and tables, that may constitute the legal document associated to the specification;
- A simulatable *semi-formal* system description: this artifact might be considered as a very useful complement that might be made available to the developers for checking their correct understanding of the system to be developed;
- *Formal* verifiable specifications, allowing the developers to possibly exploit these models for “correct-by-construction” code generation, and allowing the IMs to maintain, further verify, and possibly improve the system specification itself;
- A *set of tests* generated and successfully applied for the analysis of the various models, that can provide developers with guidance and early verification for the testing of the ongoing product development.

4.1 The Architecture of the 4SECURail Demonstrator

There are four points that directly affect the definition of the architecture of the demonstrator: (i) how the semi-formal models describing the system requirement specification are constructed for being analysed, (ii) how the simulatable models of the system are constructed, (iii) how the formal models of the system are constructed and verified, (iv) how the case study selected for the exercising the demonstrator may affect its architecture. In the remainder of this section we will discuss these four points.

Specification with Standard Notations. It is important to adopt as reference for the demonstrator a standardised notation for systems specification, considering the indications of the EULYNX⁸ and X2Rail projects, which have chosen UML/SysML diagrams (in particular their behavioural state machines and sequence diagrams). The ideal approach to system specification should rely on an advanced support framework allowing to construct diagrams that are clear, graphically appealing, rich of content and possibly interactive. Starting from these, interactive simulation to explore the possible non-deterministic alternatives present in the behaviour would be possible, allowing the formal verification of system properties. Unfortunately, this ideal approach is still far from the current state of the art. In practice, if we really want to construct diagrams that are clear, graphically appealing and rich of content, it is necessary to make use of specific drawing-oriented tools (e.g., in ASTRail, the Graphviz⁹ Graph Visualization Software has been used for this purpose) that do not support simulation and verification. Instead, diagrams automatically generated by UML/SysML-based frameworks are often of a not sufficient graphical quality and may not contain all the useful detailed information (e.g., the abstract events that relate a system transition to one or more system requirements). At the same time, however, they may be directly used to perform simulation and verification. The use of UML/SysML-based frameworks allows the progress from system design to code generation in a rather smooth way as well. This is usually of interest to developers but of less interest to the point of view of IMs. It is therefore likely, unless more experience comes out from the actual demonstrator experimentation, that a graphical SysML design will be adopted in our demonstrator without any predetermined relation with a specific UML/SysML-based framework.

Frameworks for Simulatable Modelling. As described above, the UML/SysML state diagrams descriptions might be exploited in the demonstrator not only as graphical designs with documentation purposes, or as a basis for translation into formal verifiable notations, but also as simulatable models suitable for experimenting the actual system behaviour. This requires the exploitation of much more complex (to learn, to use, to acquire) frameworks supporting execution and simulation of composite systems based on interacting state machines. The survey on semi-formal tools presented in X2Rail-2 deliverable D5.1 [38] indicates as preferred frameworks for system simulation the following ones:

- PTC Integrity Modeler (now Windchill Modeler SysSim)¹⁰;
- Sparx Systems Enterprise Architect¹¹;
- No Magic Cameo Systems Modeller (now Dassault 3DS Cameo Systems Modeller)¹².

⁸ <https://eulynx.eu/>.

⁹ <https://www.graphviz.org/>.

¹⁰ <https://www.ptc.com/en/products/plm/plm-products/windchill/modeler/sysim>.

¹¹ <https://sparxsystems.com/products/ea/index.html>.

¹² <https://www.nomagic.com/products/cameo-systems-modeller>.

Although we intend to follow this indication, at the current stage of the 4SECU-Rail project, we still need to acquire some hands-on initial experience on the chosen case study to be able to select one of these tools.

In the context of the 4SECUrail demonstrator, the exploitation of a framework allowing to directly simulate the designed behavioural models, in agreement with the official OMG fUML semantics [47], it would allow to ensure that the designed graphical models actually reflect the expected system behaviour in an unambiguous way.

Formal Verification by Model Checking. One of the project's main goals is to transform these standard UML/SysML designs, whatever supporting tool is chosen, into verifiable formal models. Theorem proving and model checking can be considered the two most used approaches to system verification, also in railway related contexts [3, 5, 15, 28]. However, theorem proving, for instance as supported by Atelier B, fits better a specification refinement process that guides the correct-by-construction generation of code starting from an initial formal design. Theorem proving moreover scales well to infinite state systems and can help identify inductive properties. Model checking instead fits better a model-based approach in which the behaviour of a simulatable design is explored and exhaustively verified. In 4SECUrail we follow the model-checking approach, partly because we are not interested in code generation. We will take advantage of the experience gained within the ASTRail project [2], where UML state machine descriptions were translated into Event-B state machines and subsequently analysed and verified by model checking with the ProB tool¹³ [7].

ProB is an animator and model checker for the B-Method. It allows animation of many B specifications, and can be used to systematically check a specification for a range of errors. ProB is one of the tools also recommended by X2Rail-2 for formal verifications. Some of the reasons for the successful experience of its use in the ASTRail project, which suggest to reuse it in 4SECUrail as well, are the following: (i) it is a free, open source product whose code is distributed under the EPL v1.0 license¹⁴; (ii) it is actively maintained and commercial support is available from Formal Mind¹⁵; (iii) it runs on Linux, Windows, and MacOS environments; (iv) it has several nice, very usable graphical interfaces, but it can also be used from the command line; (v) it is well integrated in the B/Event-B ecosystem (Rodin, Atelier B, iUML, B Toolkit); (vi) it allows construction, animation and visualisation of non-deterministic systems; (vii) it allows formal verification through different techniques like constraint solving, trace refinement checking, and model checking.

Instead, the following are some known weak points of the use of ProB [1]. (i) It does not allow the explicit modelling of multiple mutually interacting state machines. The only way to achieve that is to merge all the separate machines into a global one. (ii) Event-B state machines are different from UML/SysML state machines. At the current state of the art several proposals of translations from

¹³ <https://www3.hhu.de/stups/prob/>.

¹⁴ <http://www.eclipse.org/org/documents/epl-v10.html>.

¹⁵ <http://www.formalmind.com/>.

UML to ProB state machines have been made, but as far as we know, no industrially usable product currently supports that mapping. (iii) Model checking does not support compositional approaches based on bisimulation equivalences which are congruences with respect to parallel composition operations. In simpler words, the verification approach does not scale when the system is composed by many interacting asynchronous state machines.

Modelling the behaviour of a system through the design of a single state machine has the advantage that this design can often be translated into the notations supported by formal verification frameworks with a reasonable effort. However, if we have to verify properties that depend on the behaviour of more interacting asynchronous systems, the situation becomes more difficult. If the components are not too complex, or not too many, a possibility is to merge all of them into a unique “global” system modelled again as a single state machine. Increasing the complexity and the number of components raises the state-space explosion problem.

One solution is to constrain the verification to a not full, but rich set of scenarios. That is, verifying the system under reasonable assumptions (e.g., absence of fatal errors in certain components, only one/two/three trains moving from one RBC to another, limited presence of communication errors, just to mention some). The other solution is to exploit alternative formal notations, more oriented towards the design and verification of asynchronous interacting systems and supported by specialised theoretical basis, such as process algebras.

We are unable at the current time to evaluate the overall final complexity of the chosen case study, and whether model checking within the ProB framework will be sufficient for its formal verification. In any case our approach does not prevent the experimentation with alternative translations towards verification engines more oriented to the analysis of “parallel asynchronous systems” (e.g., mCRL2¹⁶ [14, 20], CADP¹⁷ [18], FDR4¹⁸ [31]), in the style of [36].

The overall execution flow embedding the three points discussed above is represented in Fig. 4.

The Case Study. The fourth point concerns the case study chosen to test the 4SECURail demonstrator: the RBC/RBC protocol, as specified by the UNISIG documents RBC/RBC Handover [52] and Safe Communication Interface [51].

In the ERTMS/ETCS train control system, a Radio Block Center (RBC) is responsible for controlling the separation of trains on the part of a line under its supervision. A handover procedure is needed to manage the interchange of train control supervision between two neighbour RBCs: when a train is approaching the end of the area supervised by one handing over RBC, an exchange of information with the accepting RBC takes place to manage the transaction of responsibilities. Since the two neighbouring RBCs may have been manufactured by different providers, the RBC/RBC interface is a typical product where development pro-

¹⁶ <https://www.mcrl2.org/>.

¹⁷ <https://cadp.inria.fr/>.

¹⁸ <https://cocotec.io/fdr/index.html>.

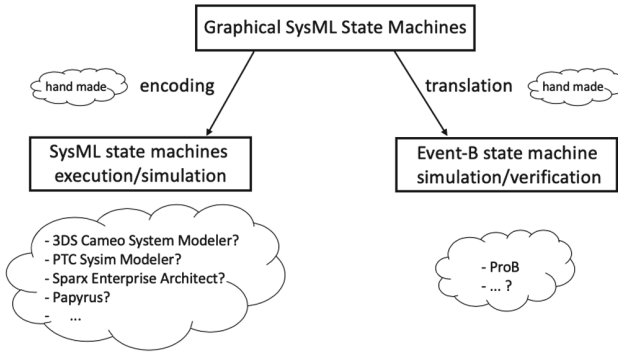


Fig. 4. Execution flow of the demonstrator prototype

cesses of different supplier meet, and is therefore an optimal choice to investigate how natural language specification may create the possibility of diverging interpretations, leading to interoperability issues. Being UNISIG SUBSET-039 and SUBSET-098 already consolidated standards, the overall structure of our demonstrator process will reflect the second point of view of those described in Sect. 4 (second case) and illustrated in Fig. 4. This is the case of the formal methods demonstrator process used for just analysing, verifying, and possibly improving an already existing standard specification.

In our particular architecture, being the input requirements an already stable official UNISIG standard, we will not rewrite it using again a natural language notation, even in the case that the rewriting could appear as more precise or complete. We can however complement it with annotations, if found useful, and/or enrich it with further artifacts developed with the demonstrator process, such as SysML models, animatable modes, formal model, test cases, and the required cross references among these components.

4.2 Input for the Cost/benefit Analysis and Learning Curve Evaluation

During the experimentation of our demonstrator process with its application to the selected case study, it will be important to assess as much data as possible about the costs of our approach. These costs might be related to actual monetary costs incurred (like the academic/research costs for acquiring some tool licence), or costs not actually incurred, but meaningful for the cost/benefit analysis (like the cost of full commercial licence for the same tool, or the cost for commercial support and training even if not activated, or cost of licence for alternative tools with respect to the ones used in the demonstrator). These costs might also be measured in terms of Man Month efforts and put in relation to the effort needed to learn a specific tool and methodology (learning curve), or to the time/effort needed to generate the animatable SysML specification, to generate the formally

verifiable models, to select, design and perform the verifications of the properties of interest, to maintain the various model well synchronized.

It will also be important to put in evidence the differential of the cost associated with the demonstrator between the cases with-or-without the exploitation of formal methods. This final exercise of the consolidated demonstrator will be the basis for studying the cost/benefit analysis of the approach and the evaluation of the learning curve for the use of the selected methodologies and tools.

5 The Adoption of a Standard Notation

UML is a standardised modeling language consisting of an integrated set of graphical diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems [53]. UML, in its SysML version, has been adopted also in the EULYNX project within its underlying methodology for the development of standard interfaces. A detailed analysis of this approach is well described in [13]. Graphical designs do actually often convey information to the reader with a wider band than just text, and require less effort in the reader for receiving it. However, a textual representation readable/writeable by humans is equally important for the simpler way in which it can be produced, shared, translated, modified, and communicated. We believe that both kinds of representation should be made available, and they should be and remain in synch.

It is also important for the designer to be able to simulate the UML behavioural models (e.g., state machines) to obtain initial feedback on the correctness of the design with respect to the intended requirements. Otherwise models risk being precise, but wrong. A prerequisite for a reasonable introduction of UML as reference notation inside a formal methods demonstrator process is that the meaning of the UML designs shall not be ambiguous or uncertain. Since its origins, this has been recognised as a major problem for some of the behavioural diagrams of UML like state machines. The main known problems with this behavioural notation are in fact: uncertainties in the semantics, absence of standard action language, and lots of implementation freedom (cf., e.g., [25, 49]).

Several studies and proposals have been conducted in the recent years with the goal of associating a formal semantics to the UML behavioural diagrams (cf., e.g., [12, 19, 37]), but none of these actually succeeded in solving the problems. An important step forward to overcome this problem has been achieved by the OMG (Object Management Group) with the standardisation of fUML (*Foundational Subset for Executable UML Models*) [47], which is also associated with an official reference implementation [40]. This definition of fUML is complemented with the definition of textual syntax for its action language *Alf* [46], and by the definition of PSCS (*Precise Semantics of UML Composite Structure*) [43]. The purpose of this fUML effort is the definition of an initial subset of UML that is free from the semantic uncertainties affecting the full standard and that might define a rigorous model of computation for the UML behavioural diagrams.

The remaining limits of this effort are that the fUML definition is still described in natural language, and that the “reference implementation” (that might play the role of unambiguous operational semantics) is currently being implemented only with respect to activity diagrams [40]. The Alf definition itself, when considered in conjunction with the state machine notation, is currently defined just through an “Informative Annex” [46] with no normative role. Within the demonstrator process, UML can play three different roles: (i) as complementary graphical *documentation* of specific aspects of the system requirements definition; (ii) as a direct notation for the execution and *simulation* of system models; (iii) as *baseline for translations* towards other formal notations supported by strong verification capabilities.

The use of UML for system *design* and *documentation* is supported by an extremely rich set of tools. If we were interested in the generation of diagrams for complementing the natural language description of a system, we might find it useful to use UML tools exploiting more immediate and user-friendly textual encoding able to automatically generate their corresponding diagrams. Support for the use of UML for *execution/simulation* of the system behaviour is limited.

None of the “industry-ready” UML tools allow direct *verification* of behavioural models; as far as we know, only a few academic prototypes have been developed for this purpose (e.g., UMC¹⁹ [4]). Therefore, we are left with the possibility of performing the translation from the UML models into other formal notations supported by verification frameworks. The literature reports numerous translation attempts [8–11, 16, 17, 32–35, 41, 45, 48, 50, 54, 55], but none of them seems to be well supported and integrated inside “industry-ready” UML frameworks.

Given the focus on formal methods of the 4SECURail demonstrator, the major interest is in the possibility of translating UML for verification, although using UML for (*documentation* and *simulation*) may play a relevant role inside the demonstrator. From this point of view our preferred choice would be the use of an even stricter subset of the fUML state machine diagrams, defining a very simple state machine structure that would allow a direct translation into the main formalisms adopted by verification and simulation tools, such as Event-B²⁰ [1], LNT [30], and Uppaal²¹ [6].

Concluding this section, the criteria that will be applied for selecting specific UML/SysML tools can be summarised as:

- unambiguity and standard quality of the supported notations;
- openness of the framework, i.e., how easy it is to import/export/translate the notations versus other frameworks;
- usability of the tool user interface;
- degree of support for non-deterministic aspects in the design and
- degree and cost of support and training by the tool providers.

¹⁹ <http://fmt.isti.cnr.it/kandisti/>.

²⁰ <http://www.event-b.org/>.

²¹ <http://www.uppaal.org/>.

6 Conclusions and Future Work

We have defined the rationale and the choices performed in terms of structure, methods, and tools selection, for the definition of a (semi-)formal software development process (demonstrator) targeted to the construction of clear, rigorous, and verifiable system specifications. In particular, two important issues deserve a specific analysis and discussion: (i) the clarification of the usefulness of formal methods from the point of view of the Infrastructure Managers, (ii) the role that the semi-formal SysML notation should play within our formal methods demonstrator process.

This defined process will be exercised for the specification and analysis of the identified case study fragment. After possible revision due to the experience gained during this first exercising of the demonstrator, a consolidated version will be used for the analysis and verification of the full case study. This final exercising of the consolidated demonstrator will be the basis for the study of the cost/benefit analysis of the approach and the evaluation of the learning curve for the use of the selected methodologies and tools.

Acknowledgements. This work has been partially funded by the 4SECUrail project. The 4SECUrail project received funding from the Shift2Rail Joint Undertaking under the European Union’s Horizon 2020 research and innovation programme under grant agreement No 881775 in the context of the open call S2R-OC-IP2-01-2019, part of the “Annual Work Plan and Budget 2019”, of the programme H2020-S2RJU-2019. The content of this paper reflects only the authors’ view and the Shift2Rail Joint Undertaking is not responsible for any use that may be made of the included information.

We also would like to thank the Italian MIUR PRIN 2017FTXR7S project IT MaT-TerS (Methods and Tools for Trustworthy Smart Systems).

References

1. Abrial, J.R.: *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, Cambridge (2010)
2. ASTRail Deliverable D4.3: Validation Report. <http://astrail.eu/download.aspx?id=d7ae1ebf-52b4-4bde-b25e-ae251fd906df>
3. Basile, D., et al.: On the industrial uptake of formal methods in the railway domain. In: Furia, C.A., Winter, K. (eds.) IFM 2018. LNCS, vol. 11023, pp. 20–29. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98938-9_2
4. ter Beek, M.H., Fantechi, A., Gnesi, S., Mazzanti, F.: A state/event-based model-checking approach for the analysis of abstract system properties. *Sci. Comput. Program.* **76**(2), 119–135 (2011). <https://doi.org/10.1016/j.scico.2010.07.002>
5. ter Beek, M.H., et al.: Adopting formal methods in an industrial setting: the railways case. In: ter Beek, M.H., McIver, A., Oliveira, J.N. (eds.) FM 2019. LNCS, vol. 11800, pp. 762–772. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30942-8_46
6. Behrmann, G., et al.: UPPAAL 4.0. In: *Proceedings of the 3rd International Conference on the Quantitative Evaluation of SysTems (QEST 2006)*, pp. 125–126. IEEE (2006). <https://doi.org/10.1109/QEST.2006.59>

7. Bendisposto, J., et al.: ProB 2.0 tutorial. In: Butler, M., Hallerstede, S., Waldén, M. (eds.) Proceedings of the 4th Rodin User and Developer Workshop. TUCS Lecture Notes, Turku Centre for Computer Science (2013)
8. Berglehner, R., Rasheeq, A.: An approach to improve SysML railway specification using UML-B and EVENT-B. In: Poster at the 3rd International Conference on Reliability, Safety, and Security of Railway Systems: Modelling, Analysis, Verification, and Certification (RSSRail 2019) (2019). <https://doi.org/10.13140/RG.2.2.21925.45288>
9. Bernardi, S., et al.: Enabling the usage of UML in the verification of railway systems: the DAM-rail approach. *Rel. Eng. Syst. Saf.* **120**, 112–126 (2013). <https://doi.org/10.1016/j.res.s.2013.06.032>
10. Besnard, V., Brun, M., Jouault, F., Teodorov, C., Dhaussy, P.: Unified LTL verification and embedded execution of UML models. In: Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MoDELS 2018), pp. 112–122. ACM (2018). <https://doi.org/10.1145/3239372.3239395>
11. Bhaduri, P., Ramesh, S.: Model Checking of Statechart Models: Survey and Research Directions. CoRR cs.SE/0407038 (2004). <http://arxiv.org/abs/cs.SE/0407038>
12. Broy, M., Crane, M.L., Dingel, J., Hartman, A., Rumpe, B., Selic, B.: 2nd UML 2 semantics symposium: formal semantics for UML. In: Kühne, T. (ed.) MODELS 2006. LNCS, vol. 4364, pp. 318–323. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-69489-2_39
13. Bui, N.L.: An analysis of the benefits of EULYNX-style requirements modeling for ProRail. Ph.D. thesis, Technische Universiteit Eindhoven (2017). https://research.tue.nl/files/91220589/2017_09_28_ST_Bui_L.pdf
14. Bunte, O., et al.: The mCRL2 toolset for analysing concurrent systems. In: Vojnar, T., Zhang, L. (eds.) TACAS 2019. LNCS, vol. 11428, pp. 21–39. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17465-1_2
15. Butler, M., et al.: The first twenty-five years of industrial use of the B-method. In: ter Beek, M.H., Ničković, D. (eds.) FMICS 2020. LNCS, vol. 12327, pp. 189–209. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58298-2_8
16. Caltais, G., Leitner-Fischer, F., Leue, S., Weiser, J.: SysML to NuSMV model transformation via object-orientation. In: Berger, C., Mousavi, M.R., Wisniewski, R. (eds.) CyPhy 2016. LNCS, vol. 10107, pp. 31–45. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-51738-4_3
17. Chen, J., Cui, H.: Translation from adapted UML to Promela for CORBA-based applications. In: Graf, S., Mounier, L. (eds.) SPIN 2004. LNCS, vol. 2989, pp. 234–251. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24732-6_17
18. Coste, N., Garavel, H., Hermanns, H., Lang, F., Mateescu, R., Serwe, W.: Ten years of performance evaluation for concurrent systems using CADP. In: Margaria, T., Steffen, B. (eds.) ISoLA 2010. LNCS, vol. 6416, pp. 128–142. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16561-0_18
19. Crane, M.L., Dingel, J.: UML vs. classical vs. RHAPSODY statecharts: not all models are created equal. In: Briand, L., Williams, C. (eds.) MODELS 2005. LNCS, vol. 3713, pp. 97–112. Springer, Heidelberg (2005). https://doi.org/10.1007/11557432_8
20. Cranen, S., et al.: An overview of the mCRL2 toolset and its recent advances. In: Piterman, N., Smolka, S.A. (eds.) TACAS 2013. LNCS, vol. 7795, pp. 199–213. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36742-7_15

21. European Committee for Electrotechnical Standardization: CENELEC EN 50128 – Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems, June 2011. <https://standards.globalspec.com/std/1678027/cenelec-en-50128>
22. Fantechi, A.: Twenty-five years of formal methods and railways: what next? In: Counsell, S., Núñez, M. (eds.) SEFM 2013. LNCS, vol. 8368, pp. 167–183. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05032-4_13
23. Fantechi, A., Ferrari, A., Gnesi, S.: Formal methods and safety certification: challenges in the railways domain. In: Margaria, T., Steffen, B. (eds.) ISoLA 2016. LNCS, vol. 9953, pp. 261–265. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47169-3_18
24. Fantechi, A., Fokkink, W., Morzenti, A.: Some trends in formal methods applications to railway signaling. In: Gnesi, S., Margaria, T. (eds.) Formal Methods for Industrial Critical Systems: A Survey of Applications, chap. 4, pp. 61–84. Wiley (2013). <https://doi.org/10.1002/9781118459898.ch4>
25. Fecher, H., Schönborn, J., Kyas, M., de Roeper, W.-P.: 29 new unclarities in the semantics of UML 2.0 state machines. In: Lau, K.-K., Banach, R. (eds.) ICFEM 2005. LNCS, vol. 3785, pp. 52–65. Springer, Heidelberg (2005). https://doi.org/10.1007/11576280_5
26. Ferrari, A., et al.: Survey on formal methods and tools in railways: the ASTRail approach. In: Collart-Dutilleul, S., Lecomte, T., Romanovsky, A. (eds.) RSSRail 2019. LNCS, vol. 11495, pp. 226–241. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-18744-6_15
27. Ferrari, A., Fantechi, A., Gnesi, S., Magnani, G.: Model-based development and formal methods in the railway industry. *IEEE Softw.* **30**(3), 28–34 (2013). <https://doi.org/10.1109/MS.2013.44>
28. Ferrari, A., Mazzanti, F., Basile, D., ter Beek, M.H., Fantechi, A.: Comparing formal tools for system design: a judgment study. In: Proceedings of the 42nd International Conference on Software Engineering (ICSE), pp. 62–74. ACM (2020). <https://doi.org/10.1145/3377811.3380373>
29. Garavel, H., ter Beek, M.H., van de Pol, J.: The 2020 expert survey on formal methods. In: ter Beek, M.H., Ničković, D. (eds.) FMICS 2020. LNCS, vol. 12327, pp. 3–69. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58298-2_1
30. Garavel, H., Lang, F., Serwe, W.: From LOTOS to LNT. In: Katoen, J.-P., Langerak, R., Rensink, A. (eds.) ModelEd, TestEd, TrustEd. LNCS, vol. 10500, pp. 3–26. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68270-9_1
31. Gibson-Robinson, T., Armstrong, P.J., Boulgakov, A., Roscoe, A.W.: FDR3: a parallel refinement checker for CSP. *Int. J. Softw. Tools Technol. Transf.* **18**(2), 149–167 (2016). <https://doi.org/10.1007/s10009-015-0377-y>
32. Grumberg, O., Meller, Y., Yorav, K.: Applying software model checking techniques for behavioral UML models. In: Giannakopoulou, D., Méry, D. (eds.) FM 2012. LNCS, vol. 7436, pp. 277–292. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32759-9_25
33. Hvid Hansen, H., Ketema, J., Luttik, B., Mousavi, M.R., van de Pol, J., dos Santos, O.M.: Automated verification of executable UML models. In: Aichernig, B.K., de Boer, F.S., Bonsangue, M.M. (eds.) FMCO 2010. LNCS, vol. 6957, pp. 225–250. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25271-6_12
34. Jussila, T., Dubrovin, J., Junttila, T., Latvala, T., Porres, I.: Model checking dynamic and hierarchical UML state machines. In: Proceedings of the 3rd International Workshop on Model Development, Validation and Verification (MoDeVa 2006), pp. 94–110. University of Queensland (2006)

35. Knapp, A., Merz, S., Rauh, C.: Model checking timed UML state machines and collaborations. In: Damm, W., Olderog, E.-R. (eds.) FTRTFT 2002. LNCS, vol. 2469, pp. 395–414. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45739-9_23
36. Lang, F., Mateescu, R., Mazzanti, F.: Sharp congruences adequate with temporal logics combining weak and strong modalities. TACAS 2020. LNCS, vol. 12079, pp. 57–76. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45237-7_4
37. Liu, S., et al.: A formal semantics for complete UML state machines with communications. In: Johnsen, E.B., Petre, L. (eds.) IFM 2013. LNCS, vol. 7940, pp. 331–346. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38613-8_23
38. Löfving, C., Borälv, A: X2Rail-2 Deliverable D5.1, Formal Methods (Taxonomy and Survey), Proposed Methods and Applications, May 2018. <https://projects.shift2rail.org/download.aspx?id=b4cf6a3d-f1f2-4dd3-ae01-2bada34596b8>
39. Mazzanti, F., Ferrari, A., Spagnolo, G.O.: Towards formal methods diversity in railways: an experience report with seven frameworks. Int. J. Softw. Tools Technol. Transf. **20**(3), 263–288 (2018). <https://doi.org/10.1007/s10009-018-0488-3>
40. ModelDriven: The fUML Reference Implementation. <https://github.com/ModelDriven/fUML-Reference-Implementation/blob/master/README.md>
41. Ober, I., Graf, S., Ober, I.: Validation of UML models via a mapping to communicating extended timed automata. In: Graf, S., Mounier, L. (eds.) SPIN 2004. LNCS, vol. 2989, pp. 127–145. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24732-6_9
42. Object Management Group: Unified Modelling Language, December 2017. <https://www.omg.org/spec/UML/About-UML/>
43. Object Management Group: Precise Semantics of UML Composite Structure (PSCS), March 2018. <https://www.omg.org/spec/PSCS/1.1/PDF>
44. Object Management Group: OMG Systems Modeling Language (OMG SysML), November 2019. <http://www.omg.org/spec/SysML/1.6/>
45. Oliveira, R., Dingel, J.: Supporting model refinement with equivalence checking in the context of model-driven engineering with UML-RT. In: Burgueño, L., et al. (eds.) Proceedings of the 20th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2017) – Satellite Events. CEUR Workshop Proceedings, vol. 2019, pp. 307–314. CEUR-WS.org (2017). http://ceur-ws.org/Vol-2019/modevva_2.pdf
46. OMG: Action Language for Foundational UML (Alf) – Concrete Syntax for a UML Action Language, July 2017. <https://www.omg.org/spec/ALF/1.1>
47. OMG: Semantics of a Foundational Subset for Executable UML Models (fUML), December 2018. <https://www.omg.org/spec/FUML/1.4/PDF>
48. Pétin, J.F., Evrot, D., Morel, G., Lamy, P.: Combining SysML and formal methods for safety requirements verification. In: Proceedings of the 22nd International Conference on Software & Systems Engineering and their Applications (ICSSEA 2010) (2010). <https://hal.archives-ouvertes.fr/hal-00533311/document>
49. Simons, A.J.H., Graham, I.: 30 things that go wrong in object modelling with UML 1.3. In: Kilov, H., Rumpe, B., Simmonds, I. (eds.) Behavioral Specifications of Businesses and Systems. SECS, vol. 523, pp. 237–257. Springer, Heidelberg (1999). https://doi.org/10.1007/978-1-4615-5229-1_17
50. Snook, C., Savicks, V., Butler, M.: Verification of UML models by translation to UML-B. In: Aichernig, B.K., de Boer, F.S., Bonsangue, M.M. (eds.) FMCO 2010. LNCS, vol. 6957, pp. 251–266. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25271-6_13

51. UNISIG: RBC-RBC Safe Communication Interface – SUBSET-098, February 2012. https://www.era.europa.eu/sites/default/files/filesystem/ertms/ccs_tsi_annex_a-_mandatory_specifications/set_of_specifications_3_etcs_b3_r2_gsm-r_b1/index063-_subset-098_v300.pdf
52. UNISIG: FIS for the RBC/RBC Handover – SUBSET-039, December 2015. https://www.era.europa.eu/sites/default/files/filesystem/ertms/ccs_tsi_annex_a-_mandatory_specifications/set_of_specifications_3_etcs_b3_r2_gsm-r_b1/index012-_subset-039_v320.pdf
53. Visual Paradigm: What is Unified Modeling Language (UML)?. <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/>
54. Yeung, W.L., Leung, K.R.P.H., Wang, J., Dong, W.: Improvements towards formalizing UML state diagrams in CSP. In: Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC 2005), pp. 176–184. IEEE (2005). <https://doi.org/10.1109/APSEC.2005.70>
55. Zhang, S.J., Liu, Y.: An automatic approach to model checking UML state machines. In: Proceedings of the 4th International Conference on Secure Software Integration and Reliability Improvement (SSIRI-C 2010), pp. 1–6. IEEE (2010). <https://doi.org/10.1109/SSIRI-C.2010.11>