Comparing Formal Tools for System Design: a Judgment Study

Alessio Ferrari
Franco Mazzanti
Davide Basile
Maurice H. ter Beek
ISTI–CNR
Pisa, Italy
firstname.lastname@isti.cnr.it

Alessandro Fantechi
University of Florence
Florence, Italy
alessandro.fantechi@unifi.it

ABSTRACT

Formal methods and tools have a long history of successful applications in the design of safety-critical railway products. However, most of the experiences focused on the application of a single method at once, and little work has been performed to *compare* the applicability of the different available frameworks to the railway context. As a result, companies willing to introduce formal methods in their development process have little guidance on the selection of tools that couldfi t their needs. To address this goal, this paper presents a comparison between 9 different formal tools, namely Atelier B, CADP, FDR4, NuSMV, ProB, Simulink, SPIN, UMC, and UPPAAL SMC. We performed a judgment study, involving 17 experts with experience in formal methods applied to railways. In the study, part of the experts were required to model a railway signaling problem (a moving-block train distancing system) with the different tools, and to provide feedback on their experience. The information produced was then synthesized, and the results were validated by the remaining experts. Based on the outcome of this process, we provide a synthesis that describes *when* to use a certain tool, and *what* are the problems that may be faced by modelers. Our experience shows that the different tools serve different purposes, and multiple formal methods are required to fully cover the needs of the railway system design process.

CCS CONCEPTS

• General and reference → Empirical studies; **• Software and its engineering → Formal methods**; **System modeling languages**.

KEYWORDS

formal methods, formal tools, empirical software engineering, judgment study, empirical formal methods, railway, moving-block system, formal methods diversity, human aspects of formal design

ACM Reference Format:
Alessio Ferrari, Franco Mazzanti, Davide Basile, Maurice H. ter Beek, and Alessandro Fantechi. 2020. Comparing Formal Tools for System Design: a Judgment Study . In *42nd International Conference on Software Engineering (ICSE '20), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3377811.3380373

1 INTRODUCTION

During the last decades, formal methods and tools have been widely applied to the development of railway systems (cf., e.g., [1, 3, 5–7, 9, 11, 12, 16, 19, 23, 28, 29, 37, 40, 42, 43, 45, 49–51, 53, 54, 59, 60, 62]). Notable examples are the usage of the B method for developing railway signaling systems in France (e.g., Line 14 of the Paris Métro and the driverless Paris–Roissy Airport shuttle [1]) and that of Simulink for formal model-based development and code generation in the development of the Metrô Rio ATP system [29]. Many projects were carried out, often in collaboration with national railway companies, for the verification of interlocking systems, which are wayside platforms controlling signals and switches in a railway yard [16, 18, 40, 43, 59, 60, 62].

Despite this long tradition, it cannot yet be said that a single method or tool has emerged as holistic solution for railway development. Thus, railway companies willing to adopt formal methods, which are highly recommended according to the highest safety integrity levels [25], are offered little to no guidance on the selection of the appropriate tools that bestfi t their needs. Indeed, most of the literature on formal methods in railways has focused on the application of a single formal method, and few researchers have addressed the problem of comparing multiple ones. Notably, Mazzanti *et al.* [52] introduces formal methods diversity, with the objective of replicating the same design with multiple tools, while Haxthausen *et al.* [39] compares two methods for the verification of a prototypical interlocking system and advocates further research in comparing formal methods.

The current paper extends the literature by performing a judgment study [57] involving 17 different subjects with experience in formal methods and railways, and concerning 9 formal tools. The study consists of two stages: an evaluation phase and an assessment phase. In the evaluation phase, three of the subjects acted as designers and were involved in an initial tool trial, in which they were required to model the requirements of a railway problem, viz., a moving-block signaling system, and report about lessons learned. Two subjects acted as analysts, and interpreted the insights provided by the designers to produce a categorization of the different tools. In the assessment phase, the categorization was validated by the remaining 12 subjects, who were required to provide confirmation or rebuttal of the results produced based on their knowledge of the tools. Thefi nal categorization can be used to guide the selection of tools in railways. The results indicate that different tools are appropriate for different contexts, characterized by the development phase (e.g., requirements, design), the type of task to address (e.g., simulation, modeling, formal verification), the type of system (e.g., single system or system-of-systems) and the typical user's background that facilitates the usage of the tool. Furthermore, we also show that several issues should be considered when choosing a certain tool, such as the presence of an intuitive GUI, the support for concurrent systems, and the possibility to define hierarchical designs. While a holistic formal methods-based solution for railway development does not exist, a combination of existing tools may provide extensive coverage of the design process.

Outline. Sect. 2 presents the background of the study. Sect. 3 presents the study design and Sect. 4 presents the results. Sect. 5 synthezises main observations, and Sect. 6 concludes the paper.

2 BACKGROUND

This section provides a brief background on formal and semi-formal methods, useful to understand the rest of the paper, and it reports related work on applications of formal methods to railway problems and comparison of formal tools. Furthermore, we introduce the context of the research project in which this work was developed.

2.1 Formal Methods

The term *formal methods* encompasses a series of mathematics-based techniques for the specification, analysis and verification of complex systems [61]. Formal methods are normally supported by tools, oriented to facilitate the specification of systems, and to check the correctness of the specification by means of formal verification. Formal verification is supported by two main families of techniques: theorem proving [13, 14, 33] and model checking [4, 20, 21]. With theorem proving the formal proof of correctness is given by demonstrating mathematical theorems about properties of the specified system. Model checking, on the other hand, enables the systematic exploration of the state space of the system to verify that the desired properties hold. Model checking can be explicit or symbolic, depending on the encoding of the state space. Model checking can also be probabilistic, when the specification exhibits a probabilistic behaviour [4]. Recently, also statistical model checking has been introduced [46], which combines simulation and statistical methods. Some model-checking tools, e.g., CADP [36], support also other verification techniques, such as equivalence checking, which verifies that two specifications' behaviors are equivalent. Other tools, such as FDR4 [38] and ProB [48], support refinement checking, which verifies that a certain specification is consistent with respect to a more abstract one.

As formal verification is concerned with verifying system properties, it is useful to distinguish between properties that are mainly qualitative (e.g., "in case of error, the train shall brake") and those that include quantitative aspects (e.g., "in case of error, the train shall brake within 500 meters"), and therefore require some form of *quantitative analysis.*

Semi-formal methods refer to formalisms and languages that are not considered fully 'formal', for which the semantics is not completely defined. These include the Unified Modeling Language (UML) [55] and dialects thereof, and Simulink [24]. In the remainder of the paper, we will use the term formal methods in a rather liberal way, including also cases of semi-formal ones, and we will specify that a certain tool or method is semi-formal only when appropriate.

2.2 Related Work

The railway domain is characterized by stringent safety requirements and a rigorous development process. The use of formal methods is highly recommended for platforms of the highest Safety Integrity Levels (SIL-3/4) by the CENELEC EN 50128 standard for the development of software for railway control and protection systems [25]. Indeed, formal methods and tools are widely applied to the development of railway systems. The extensive survey on applications of formal methods by Woodcock *et al.* [63], which includes a structured questionnaire submitted to the participants of 56 projects, identified the transport domain, including railways, as the one in which the largest number of projects that include applications of formal methods have been performed.

In the past, several authors have reviewed formal modeling and verification languages and tools commonly used in the railway domain (cf., e.g., [1, 15, 17, 26, 27, 34]). Bjørner [15] presents afirst, non-systematic survey of formal methods applied to railway software, and lists the main reference techniques and tools, including the B method [2] and SPIN [41], used in about 180 papers. Fantechi *et al.* [26, 27] performs a similar review, updated with new applications and indicating future challenges related to the increasing complexity of railway systems. With a focus on the B method, Abrial [1] and the book edited by Boulanger [17] focus on successful industrial applications of the method, including railway experiences at Siemens and other companies. The book edited by Flammini [34], which covers different aspects of railway system development, also dedicates two chapters to formal methods applications. Overall, these contributions demonstrate the interest of researchers and practitioners in the topic. However, no comparison between tools is performed in any of the studies. The only contributions that perform some comparison between methods are those by Mazzanti *et al.* [52] and Haxthausen *et al.* [39]. Similar to our case, the former focuses on the replication of the same design with different tools, while the latter compares two formal methodologies for interlocking system development. In both cases, the results are based on the authors' experience. Our work makes a step forward, by performing a judgment study involving 17 experts (5 authors, 12 external) and 9 tools, and paves the basis for further research on formal methods comparison.

2.3 Context

The work described in this paper is one of the outputs of a larger endeavour performed in the context of ASTRail[1] (SAtellite-based Signalling and Automation SysTems on Railways along with Formal Method and Moving Block Validation), a project funded by EU's Shift2Rail initiative[2]. This aims to increase the competitiveness of the European railway industry, in particular concerning the transition to the next generation of EU signaling systems, which will include satellite-based train positioning, moving-block distancing,

[1]astrail.eu
[2]shift2rail.org

and automatic driving. Below we briefly present the ASTRail project and the moving-block system that will be considered in our study.

The ASTRail project aims to introduce recent scientific results and methodologies as well as cutting-edge technologies from other transport sectors, in particular avionics and automotive, in the railway sector. The project leverages formal methods and tools for careful analyses of the resulting novel applications and solutions in terms of safety and performance. One of ASTRail's aims is to define a *moving-block* signaling system, according to which a safe zone around the moving train can be computed, thus optimizing the line's exploitation. For this solution to work, it requires the precise absolute location, speed, and direction of each train, to be determined by a combination of sensors: active and passive markers along the track, as well as train-borne speedometers.

As part of the ASTRail project, we are currently surveying and assessing the main formal methods and tools that are being used today in research projects on railway systems, to identify the ones that are most mature for application in the railway industry. To this end, a survey on applications of formal methods in railways was foreseen in ASTRail, followed by a judgment study in which the most mature tools, according to the survey, are evaluated by experts to provide guidance on the most appropriate tools to be used for railway development. This paper reports the results of the judgment study, while the results of the survey are reported in previous work [10, 32].

3 RESEARCH DESIGN

To study the problem of comparing different formal tools, we performed a *judgment study* [57] composed of two phases: an *evaluation* phase and an *assessment* phase (see Fig. 2, discussed in detail later). In the evaluation phase a group of three experts in formal methods and railways was required to model a railway system (viz., the moving-block system) with different tools, and, for each tool, provide a qualitative evaluation based on the experience. The data was analyzed and synthesized by two other experts, who produced a set of categories to characterize the different tools. The produced categories were used as input for the assessment phase, in which experts of each specific tool were required to validate the results. Although other empirical methodologies exist to systematically compare different tools (e.g., the DESMET method [44]), we argue that a design such as the one adopted is more appropriate for an early research phase on a complex topic as ours. Indeed, the evaluation phase is used to identify themes that typically characterize different formal tools, while the assessment phase aims to generalize the results produced in the contrived context of the evaluation phase. The presented study is *exploratory* and *interpretative* in nature, since it aims tofi nd somefi rst insights on the topic, and it is based on the participants' interpretation of their experience.

In the following, wefi rst outline the research questions, then we describe the case used during the evaluation phase, the tools considered, the characteristics of the study participants, and the experimental procedure adopted.

3.1 Research Questions

The research objective of this study is to compare different formal tools for their applicability in the railway context. To address this objective, two main research questions (RQ) are defined to drive the study of each tool:

- **RQ1:** *When is it appropriate to use a certain tool for the design of railway systems?* This question aims to understand for which purpose the tool is more appropriate and which are its main strengths.
- **RQ2:** *What issues should be considered when using a certain tool?* By answering this question we want to highlight which are the situations in which the tool is not appropriate and what are the potential hurdles that one should consider when choosing the tool.

By combining the answers to these questions, we aim to provide a comparative synthesis to have a better understanding of the applicability of the tools to the railway context.

3.2 Case for the Evaluation Phase

The case under study for the evaluation phase consists of a railway system to be designed. The chosen system to be modeled is the moving-block system. The selection is opportunistic, in that the study was performed in the context of a project in which the moving-block system was one of the primary objectives (see Sect. 2.3). In the context of the project, a high-level UML model of the moving-block system was produced by the industrial partners. Based on such a model, a set of textual requirements was defined, meant to be the primary source of information for the formal modeling activity. Here we describe the main principles and components of the moving-block system. Fig. 1 provides an overview of the system. There are three components: two on the train, and one wayside system. The train carries the Location Unit (LU) and the Onboard Unit (OBU). The wayside system is the Radio Block Centre (RBC). The location of the train is computed by the LU by means of sensor fusion algorithms, and sent to the OBU, which, in turn, sends the location to the RBC. Upon reception of a location from a train, the RBC sends a Movement Authority (MA) to the OBU. The MA indicates the space that the train can safely travel, considering the safety distance with the preceding train. There are also two temporal constraints that the OBU shall consider: the location must be updated within 5 seconds maximum; the MA must be updated within 15 seconds maximum. Whenever one of these constraints is violated, the OBU shall force the train to brake. The requirements and developed models are reported in our public repository [30].

3.3 Tools

The tools were selected amongst the top ranked ones in a survey on the application of formal methods to the railway domain, reported in recent contributions [10, 32]. The survey consisted of a systematic literature review including 114 papers, complemented with a project review based on 8 projects and a questionnaire with 44 practitioners. We selected the tools that emerged as most used in industrial railway cases[3], plus two additional tools, CADP and FDR4, chosen as representative tools for process algebras, as these are explicitly mentioned in the railway norms [25]. Below, we list the tools together with their version and a brief description.

[3]The SCADE framework, ranked among the most mature tools, could not be included in the study for licensing reasons. However, independent experiences conducted by the 5th author suggest that SCADE shares with Simulink most of the expressed results.

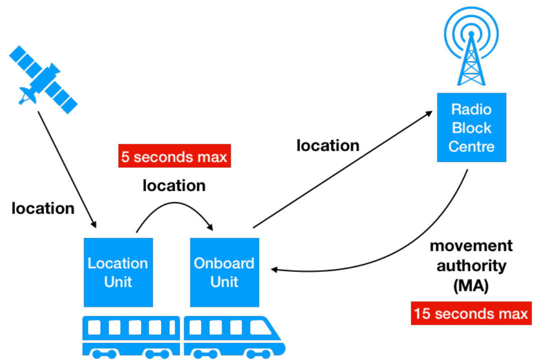

Figure 1: Overview of the moving-block system

- **Simulink** (2017b)[4]: Simulink is a commercial model-based development tool that allows the user to graphically draw diagrams of the system modeled in the form of input-output blocks. Blocks can be further refined in the form of hierarchical state machines through the Stateflow tool, included in Simulink. Simulink supports graphical simulation of the diagrams, and Simulink Design Verifier, a package included in Simulink, allows formal verification of properties. Simulink comes with several other packages, also for code generation.
- **UMC** (4.8)[5]: UMC is a verification framework for the definition, exploration, analysis, and model checking of system designs represented as a set of communicating (UML) state machines. Its current state is that of an experimental framework mostly used for teaching and research purposes. UMC is available with a free license, and directly accessible from an online server.
- **UPPAAL SMC** (4.1.4)[6]: UPPAAL SMC is a Statistical Model Checking (SMC) extension for the UPPAAL toolset, a well-known integrated tool environment for designing and analyzing real-time systems modeled as networks of (stochastic) timed automata, extended with data types. It supports the design of systems that can be modeled as a collection of non-deterministic or probabilistic processes withfi nite control structure and real-valued clocks, communicating through channels or shared variables. Both academic and commercial licenses are available.
- **Atelier B** (4.2.1)[7]: Atelier B is a theorem prover for Event B that allows the user to specify invariant properties of an abstract design developed according to the B methodology, and supports the user in performing the necessary proofs to ensure that these invariants are successfully preserved by its successive refinements. The refinements can reach a level very close to that of an actual executable program, and the framework itself allows the translation of thefinal implementations into C or Ada code. It is available in a free community edition and in a maintenance edition.

[4]https://www.mathworks.com/products/simulink.html
[5]http://fmt.isti.cnr.it/umc/V4.8/umc.html
[6]http://people.cs.aau.dk/~adavid/smc/
[7]https://www.atelierb.eu/en/

- **ProB** (1.10.2018)[8]: ProB is an animator, constraint solver, and model checker for the B Method[9]. It allows fully automatic animation of B specifications, and can be used to check a specification for a wide range of behavioral or data related errors. ProB is free to use and open source; commercial support is provided by the spin-off company Formal Mind[10].
- **NuSMV** (2.6.0)[11]: NuSMV is a reimplementation and extension of the SMV symbolic model checker. NuSMV is a tool distributed with a free open source license. Since version 2, it combines BDD-based model checking with SAT-based model checking (see [21] for details of these techniques).
- **SPIN** (6.4.9)[12]: SPIN (Simple Promela INterpreter) is an advanced and efficient on-the-fly model checker specifically targeted for the verification of LTL properties over multi-threaded software. The tool is actively maintained and available with a free open source license.
- **CADP** (2019-a)[13]: CADP (Construction and Analysis of Distributed Processes) is a verification framework for the definition and analysis of asynchronous concurrent systems. It offers interactive simulation of the system, on-the-fly verification of branching-time logic formulas, and code generation for the system components. Its integration with equivalence-checking tools enables advanced compositional verification. It also offers quantitative analysis in the form of performance evaluation [22]. CADP is actively maintained and usable with a free academic license or with a commercial license.
- **FDR4** (4.2.3)[14]: FDR4 is a verification framework that allows the user to verify refinement relations of programs written in CSP$_M$, a language that combines the operators of Hoare's CSP with a functional programming language, or tock CSP (a timed version). It exploits a compositional parallel refinement checking engine. FDR4 is available with a free academic and research license or with a commercial license.

3.4 Study Participants

The study participants involved as subjects in the evaluation phase are the authors of the paper. Altogether, their expertise covers the areas of formal and semi-formal methods, as well as railway system development. Five subjects were involved. Thefi rst three subjects, referred in the following as *designers*, performed the modeling activity. The other two (*analysts*) used the information produced to create a synthesis. The subjects are characterized as follows.

- **Semi-formal Railway Designer** (1st author): more than 10 years of experience (of which 3 years in industry) with applying semi-formal methods in railways. He used Simulink to model the problem.
- **Multi-formal Designer** (2nd author): over 20 years of experience with multiple formal methods, with specific focus

[8]https://www3.hhu.de/stups/prob
[9]http://www.methode-b.com/en/
[10]http://formalmind.com
[11]http://nusmv.fbk.eu
[12]http://spinroot.com/spin/whatispin.html
[13]https://cadp.inria.fr
[14]https://www.cs.ox.ac.uk/projects/fdr/

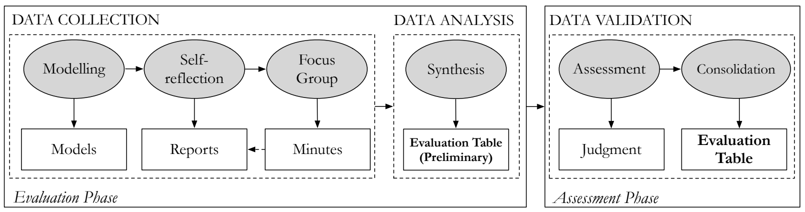

Figure 2: Overview of data collection, analysis and validation

on model checking. About 7 years of experience in applications of formal methods to railway problems. He used ProB, NuSMV, SPIN, UMC, CADP, and FDR4.

- **Probabilistic Designer** (3rd author): more than 5 years of experience with probabilistic and statistical model checking. About 3 years of experience in applications of formal methods to railway problems. He used UPPAAL SMC and Atelier B.
- **Formal Methods Analyst** (4th author): 15 years of experience with multiple formal methods, with specific focus on the application of model-checking tools, among which probabilistic and statistical variants, also to railway problems.
- **Railway and (Semi-)formal Methods Analyst** (5th author): more than 30 years of experience in formal methods and railways, with a focus on model checking.

Tools were associated to designers based on the background of the subjects, and their previous familiarity with the tool and underlying formal theory. We did not want the designers to dedicate effort to learning the theoretical foundations of the tools, but to reflect on their specific features based on the hand-on experience. The usage of the same problem to model (i.e., the case presented in Sect. 3.2) can be regarded as a *trigger* to reflect on the tools' capabilities in the specific railway context.

In the assessment phase, one or more external experts, referred to as *assessors*, were selected for each of the tools considered. The assessors were selected among the authors of papers on the application of a specific tool to a railway problem, based on the results of survey [32]. Table 1 summarizes each assessor's expertise concerning a specific tool. Two assessors were involved for a tool when designers admitted to have less confidence on their evaluation.

3.5 Data Collection, Analysis and Validation

The different steps of the study are depicted in Fig. 2. Data collection and analysis correspond to the evaluation phase, while data validation corresponds to the assessment phase. We now describe the activities performed in each phase.

- **Modeling:** Each designer started with the same material and independently developed the models for a subset of the tools. Specifically, the designers were required to interpret the graphical UML model and the requirements (see Sect. 2.3), and provide their interpretation using the languages of the tools. Furthermore, they were required to explore the capabilities offered by each tool such as, e.g., to verify certain properties, or to observe the graphical simulation capabilities of the tool. As the focus was on formal design and not on verification, full proof of requirements satisfaction was not required. We asked to test the verification capabilities of the tools by selecting some properties that the designers considered relevant. The designers did not communicate during the development, and had about ten days to develop each model. No incentive was given to the designers, as they were all involved in the ASTRail EU project, and it was part of the project's plan to model the same problem with different tools. The result of this activity is a set of models of the moving-block system [30], with some variants of the models developed when the designer considered it appropriate to experiment with multiple designs.
- **Self-reflection:** Each designer was required to reflect autonomously on their experience in using the assigned tool and to produce a written report (1–2 pages) with a motivated answer to the RQs for each considered tool, and with excerpts of the models when considered appropriate.
- **Focus Group:** While the previous phases did not involve any interaction, in this phase the three designers met for a two-stage focus group. In thefi rst stage, the designers showcased the tools with the support of the developed models, and 15 minutes were allowed for each tool's presentation. In the second stage, which lasted about three hours, each person using a certain tool was required to think aloud by answering the RQs. The other designers challenged the speaker in case of disagreement or in case more clarification was considered to be appropriate. Extensive notes were taken during the focus group and the complete results were documented in the minutes of the meeting. Furthermore, the designers were required to update their reports based on the minutes.

Table 1: Experience of the assessors with the tools

Tool	Assessor ID	Years of Experience	Industrial Projects	Railway Projects
Simulink	1	3 to 10	1 to 3	1 to 3
UMC	2	<1	1 to 3	1 to 3
UPPAAL SMC	3	3 to 10	3 to 10	1 to 3
NuSMV	4	3 to 10	1 to 3	1 to 3
	5	3 to 10	1 to 3	1 to 3
SPIN	6	>10	1 to 3	1 to 3
Atelier B	7	>10	3 to 10	3 to 10
	8	1 to 3	3 to 10	1 to 3
ProB	9	1 to 3	1 to 3	1 to 3
	10	3 to 10	1 to 3	1 to 3
CADP	11	>10	3 to 10	0
FDR4	12	>10	1 to 3	1 to 3

- **Synthesis:** Reports and models were used as input by the analysts to perform a qualitative analysis. Specifically, the subjects worked in pairs and performed a thematic analysis [58] of updated reports, to identify recurrent themes and provide a table to compare the tools. Whenever some information was considered unclear or somewhat in contrast with their knowledge, they inspected the models or consulted the designers for further clarification. The result of this activity was a preliminary evaluation table, including categories of evaluation (e.g., purpose for using the tool), and associated values (e.g., simulation, formal verification) for each tool.
- **Assessment:** The evaluation table produced in the previous phase was used as input to ask each expert of a certain tool (*assessor*) whether they agreed with the result produced about the tool. Each assessor was independently contacted via email. For each category and value specified in the table for the tool, we asked the assessors to indicate their agreement. Furthermore, we asked them to openly discuss issues that a user should consider when deciding to use the tool for system design in an industrial context.
- **Consolidation:** Whenever the assessors disagreed, we asked them to clarify the motivation of their opinion. If the clarification led to a consensus, or new themes emerged, the categorization was consolidated in afi nal evaluation table. If a consensus could not be reached, we kept all presented opinions in the categorization, specifying those for which the viewpoints of experts resulted different.

3.6 Threats to Validity

Threats to validity are discussed according to the categories of validity, reliability, and generalization outlined by Leung [47].

Validity. One of the main requirements for judgment studies is the degree of experience of the subjects involved [57]. The expertise of the designers spans through the areas of formal methods and railway system development. Therefore, their opinion on the study topic is informed, and the focus group organized helped to ensure alignment. The expertise of the designers is also fully covered by the analysts, thus ensuring the presence of a counterpart to possibly challenge the opinions produced by the designers. Finally, the assessment phase involved subjects with strong expertise in each tool, and this further strengthens the validity of thefi nal results. Threats of descriptive validity are also limited as (1) the authors of the paper are also the experts involved in the evaluation phase, so missing information would be identified; (2) all interactions with assessors were performed in written form. The main residual threat is associated to researcher bias. Although this cannot be ruled out, we argue that the involvement of the assessors mitigates this bias. We also could not avoid that the same designer used similar solutions when using different tools, for example to avoid certain design pitfalls after facing them with a certain tool. However, as the focus of the study is on the tools' features and not on the specific problem, this aspect should have limited impact on validity. Pitfalls associated to the design and occurring with a certain tool may not occur with a different one using another language.

Reliability. Four main triangulation strategies aimed to ensure the reliability of data and results: (1) the focus group allowed the participants to discuss and update the information produced in the reports, thus limiting potential tunnel vision of the designers; (2) the analysis phase was performed by two analysts, to limit potential subjectivity in the interpretation of the data; (3) the analysis phase involved interaction with the designers, to ensure correct interpretation and completeness of the information; (4) the validation phase included an iteration in which the judgment provided could be complemented with further clarifications. Besides the description of the study design and participants reported in the previous sections, we also share the initial requirements, and the produced models, to support replication [30].

Generalization. The results of the evaluation phase can be considered applicable for railway problems that are similar to the one considered, involving onboard and wayside systems, as well as temporal constraints. However, the results were consolidated based on the opinion of experts, which were unaware of the specific context that produced them. Therefore, we argue that the results can be considered sufficiently general for the railway domain, notwithstanding the inherent limitations of judgment studies due to the absence of representative sampling [57]. The results may be applicable to other domains with comparable systems' architectures, and similar safety-critical constraints (e.g., automotive, aerospace).

4 RESULTS

Table 2 reports the evaluation table resulting from the assessment phase. Thefi rst four columns are related to RQ1, while the last column addresses RQ2. Controversial aspects for which a consensus was not reached are indicated in *italic*. In relation to RQ1, four main recurrent themes were identified, which are considered as evaluation categories: development *phase*, *purpose*, railway *system* type and *facilitating user skills*. For these themes, values are specified for each tool, according to the following definitions.

- **Phase:** We distinguish four phases, namely *requirements* (R), when the tool is considered appropriate for early prototyping towards the definition of requirements; *high-level design* (H), when the tool provides proper support for high-level system design; *detailed design* (D), when the tool and its language are considered sufficiently expressive to be applied for detailed design; *implementation* (I), when the tool supports code generation.
- **Purpose:** We distinguish the following purposes: *system modeling* (SM), *simulation* (SIM), *quantitative analysis* (Q), and *formal verification* (FV) by means of *model checking* (MC), *equivalence checking* (EC), *theorem proving* (TP), or *refinement checking* (RC).
- **System:** We distinguish three system types: *single system* (S), such as a single RBC, OBU, or LU (see Sect. 2.3); *composite system* (C), such as the composition of unique instances of single systems; *system-of-systems* (SoS), such as the composition of multiple instances of single systems (e.g., multiple RBC controlling multiple trains and associated OBUs).
- **Facilitating User Skills:** These indicate the user's background that can facilitate the usage of a certain tool. Mathematical logic is assumed as required background for all the

Table 2: Evaluation table comparing the different tools

Tool	Phase	Purpose	Syst.	Facilitating User Skills	Issues
Simulink	R/D/I	SM/SIM	S	Computer engineering	commercial (T), partial parallelism (L), no architecture (L), semi-formal (L), closed format (L)
UMC	R/D	SM/FV-MC	C	UML, OO programming	prototypical (T), closed systems (L), limited size (V), limited documentation (T), limited hierarchy (L)
UPPAAL SMC	R/H	SM/SIM/Q	C	Quantitative analysis	statistical background (U), no nested temporal properties (V), maturity of theory (T), no hierarchy (L)
NuSMV	H	FV-MC	S	Symbolic model checking	specification language difficult to master (L), no GUI T), confidence with framework (U), no parallelism (L), limited hierarchy (L)
SPIN	H/D	FV-MC	C	C programming	closed systems (L), no hierarchy (L), confidence with framework (U), *limited GUI* (T)
Atelier B	H/D/I	FV-TP-RC	S	Proof theory	not fully automatic (V), *theorem-proving background* (U), no temporal properties (V), *limited parallelism* (L)
ProB	R/H	FV-MC-RC	S	Multiple FM	*limited parallelism* (L), *not design-oriented* (T)
CADP	R/H/D/I	SM/FV-MC-EC/Q	SoS	Concurrency theory	limited GUI (T), toolbox (T)
FDR4	R/H	SM/FV-RC	SoS	Process algebra	no temporal properties (V)

tools considered, except Simulink, which does not strictly require this type of competence.

It is worth mentioning that in Table 2 we report the phase, purpose, or system for which the tool has been considered more appropriate, which by no means implies that a tool is useless in other contexts.

Regarding RQ2, a set of unique themes, common to only a subset of the tools, emerged from the different data sources; they are listed as *Issues* in Table 2. These are partitioned into (L) *language*-, (U) *user*-, (T) *tool*-, and (V) *verification*-related issues.

In the following, we provide a justification for the themes that are the most characteristic of each tool[15]. To this end, we include relevant parts of the text of the updated reports, and we highlight in **bold** the theme acronym or name when the reported text justifies a certain theme. When relevant, we report observations of assessors. The goal is twofold: provide a short guided tour of each tool, and at the same time provide evidence of the identified themes. Part of the tools share some fundamental characteristics, and therefore we discuss them in groups. Specifically, we considerfi ve main groups, namely tools oriented to simulation (Simulink), to UML verification (UMC), to real-time and probabilistic aspects (UPPAAL SMC), to refinement (Atelier B, ProB), to verification of temporal logic properties of large single or composite systems (NuSMV, SPIN), and to large-scale analysis of systems-of-systems (CADP, FDR4).

4.1 Modeling for Simulation

Simulink is a model-based development tool that offers graphical modeling and provides powerful simulation capabilities as well as code generation.

RQ1: When to use tools such as Simulink? Simulink is mostly appropriate in two, rather different phases of the development, namely: (1) requirements phase (**R**), in which system prototypes are developed to support the definition of the requirements; (2) detailed design phase (**D**), in which the system model shall be close to the final implementation. Indeed, the tool supports simulation (**SIM**) in the form of animation of graphical models, which can be useful in the initial phases of the development process, to providefirst experiments, increase the confidence on the initial design, facilitate interaction with the customer, and establish the initial requirements. The simulation feature is also useful in the detailed design phase, in that it enables debugging capabilities. Furthermore, for the detailed design phase, the tool offers the code generation feature (**I**).

Given its expressive and graphical language and its focus on the implementation, Simulink is appropriate for modeling (**SM**) the internal behavior of single railway components (**S**), when these can be represented as state machines, as in the case of OBU, RBC, and LU. The statecharts notation used by Simulink—and in particular by Stateflow, the Simulink package for state machine diagrams—is generally oriented towards users with a **Computer Engineering** background. Finally, the tool is appropriate when a company wishes to have a holistic platform, covering different phases of the development—from prototyping to code to testing—, and having different packages for multiple purposes like, e.g., report generation, model-based testing.

RQ2: What issues should be considered when using Simulink? Simulink is a **commercial** tool with a licensing cost varying with the number of packages that one wishes to purchase. Therefore, the models developed can be normally read and executed solely by the tool—some porting capabilities are available, but the format is not open (**closed format**). Hence, if a company invests in this tool, and develops artifacts with it, it creates a dependable business relationship with the tool vendor. Simulink supports the modeling of different systems interacting with each other, as in the case of the moving-block system, and can support concurrent execution (i.e., parallelism) at the level of Simulink blocks. However, Stateflow state machines (see Sect. 3.3) do not support parallelism: each machine executes after the preceding one, where default precedence follows the western reading direction—left to right, top to bottom (**partial parallelism**). It should be noticed that, in Simulink there is no package for high-level architecture design (**no architecture**). This problem can be addressed by creating different architectural hierarchies with the Simulink subsystems, i.e., blocks that can contain other blocks.

4.2 UML Verification

Despite of its ambiguities and lack of rigor, UML behavioral models can be useful to communicate design skeletons in a graphical way. Even if model-based design frameworks exist that allow evolving these models into actual code (e.g., IBM Rational Software Architect[16]), UMC is one of the few tools that allow reasoning on these behavioral models in terms of temporal properties.

RQ1: When to use tools such as UMC? UMC is mostly appropriate in the requirement phase (**R**) of software development, when the stakeholders are interested in improving the confidence that the

[15]Themes that are less characteristic for a certain tool are still reported in the table.

[16]cf. https://ibm.co/2HrMT9L

graphical behavioral models on which they are working actually meet their expectations. UMC allows flexible system modeling (**SM**) in textual form, and supports step-by-step simulation of the possible system evolution, to draw an abstract graph of such possible evolution, to specify and verify branching-time temporal properties over the system behavior (**FM-MC**), and to observe detailed explanations when verification fails. Thanks to its flexible set of data structures and operations, UMC can also be used to refine a very abstract design skeleton into a prototype relatively near to a possible real implementation of the system (**D**). In UMC, a system design is seen as a closed system constituted by a *flat* (**limited hierarchy**) set of communicating UML state machines (i.e., the environment should be included as part of the design). Therefore, UMC fits well the need of verifying the behavior of interacting asynchronous systems (**C**). The use of UMC does not require a high degree of competence in formal methods, since the behavior of the model and the properties to be verified can both be specified in an easily accessible way. However, the object-oriented (OO) syntax of its input language may feel familiar to someone with experience in **OO programming** or **UML** design.

RQ2: What issues should be considered when using UMC? UMC is developed and maintained in the context of an academic project and mostly used for teaching and research purposes. UMC does not have the maturity of a commercial tool, nor any significant history of use in industrial settings (**prototypical**), and has **limited documentation**. Another aspect to be taken into consideration is that the OMG UML specification [55] is intended to cover only the description of a generic state machine: the overall behavior of a system composed by multiple machines and many aspects of the machines themselves (e.g., the precise behavior of the event queues) are left intentionally not completely specified in the specification. Therefore, UMC relies on a set of design aspects that might differ from those of other UML-based environments. Moreover, we have seen that UMC is oriented to the description of **closed systems** and it is thus not the best choice when a fragment of a system (e.g., a single component) has to be verified in isolation, in an out-of-the-loop way. UMC is an on-the-fly explicit model checker that fits very well the goal of debugging—likely wrong—system designs, but that might cause difficulties when used for validating particularly large—likely correct—system designs (**limited size**).

4.3 Modeling Real-Time and Probability

UPPAAL SMC is the only tool among the ones considered that mainly focuses on analysis of real-time and probabilistic aspects.

RQ1: When to use tools such as UPPAAL SMC? A tool oriented to timed/probabilistic aspects is useful when these quantitative aspects play an important role in the definition of the expected system properties (**Q**). It is also reasonable to imagine that, given a design specified in a functional way with other approaches, fragments of it are modeled with tools like UPPAAL SMC for a more specific time-oriented verification. In our specific case of the moving-block application, several requirements are expressed under the form of time assumptions (e.g., "OBU cycle shall be 500 ms", "the train must stop if no MA is received for 15 s"). However, these assumptions are very simple and can easily be approximated without a rigorous modeling of the flow of time (e.g., "the train must stop if no MA is received for 20 consecutive OBU cycles"). Tools like UPPAAL SMC would allow better understanding of the underlying timed/probabilistic aspects, allowing the verification of properties (when all relevant numerical values are provided) such as: "What is the probability for the train to enter in the braking state within 10 s?"; or "If the OBU-LU communications delays are in the range 10–100 ms, then what is a reasonable requirement for the response time of LU to guarantee that the OBU cycle of 500 ms is never preempted?"

RQ2: What issues should be considered when using UPPAAL SMC? While standard functional analysis methods are studied for decades and for which several industry-mature tools (like model checkers) exist, tools oriented to Statistical Model Checking (SMC) are very recent (**maturity of theory**). For example, the first version of UPPAAL SMC was released in 2014. As for classical model checkers, certification of UPPAAL SMC for the development of critical software is not available. Several problems, known to be decidable in discrete systems, cease to be so in a real-time probabilistic setting. Hence, the set of properties on which quantitative analysis can be carried out is more limited, and in particular nesting of temporal operators is not allowed (**no nested temporal properties**). Moreover, even though the formalism is closer to state machines, a good knowledge of the underlying mathematical formalisms used to specify both the model and the properties, as well as to analyze the obtained results is definitely required. Therefore, a certain specialization of developers, engineers, and other users in real-time stochastic model-based analysis is needed (**statistical background**).

4.4 Modeling for Top-Down Development

ProB and Atelier B are examples of coordinated verification environments that support the formal analysis and development of Event B specifications. An Event B specification consists essentially of the definition of a single state machine whose evolutions are triggered by external signals. The focus of the methodology is to validate the correctness of the state machine either by proving that it is a correct *refinement* of another more abstract state machine as in Atelier B, or by model checking its dynamic behavior as in ProB.

RQ1: When to use tools such as Atelier B? Atelier B is an interactive theorem prover for assessing the preservation of structural properties (in the form of invariants) on the status of an Event B model (**FV-TP**). This approach is particularly useful when a single state machine (**S**) has a complex internal status, of which the possibility to guarantee the preservation of consistency is of primary importance. The language of Atelier B uses mathematical objects (e.g., sets, existential or universal quantifiers, data types, relations, functions). This aspect has great advantages when the system to be modeled and verified can be naturally rendered as a set of logical connectives. In our case, the moving-block design under examination does not have these characteristics, since the local machine status is essentially constituted by a 'counter' variable (the maximum number of cycles that can be executed before stopping the train in absence of a MA, see Sect. 2.3) whose value should invariantly remain in the range $0 \ldots 15$. Nevertheless, in the railway context, several examples can be found for which this approach might be useful (e.g., interlocking systems [62]). Given that it belongs to the B method suites, Atelier B can be applied in most of the software life-cycle (**H**/**D**/**I**).

The strongest point of using a theorem prover is the possibility of verifying properties for systems with a potentially infinite number of states. On the other side, proving a theorem is not a completely automatic procedure and it requires several interactions with the user, who is in charge of selecting the specific strategy to prove a certain result. Nevertheless, Atelier B is equipped with a feature for trying to automatically prove certain simple properties, which does not always succeed but can often be helpful. When an Event B model is refined, Atelier B automatically generates proof obligations to be discharged by proving them. Such proof obligations mainly consist of proofs of preservation of invariants by refinement, and of the correct refinement of pre- and postconditions of events (**FV-RC**). Of course, another strong point of Atelier B is the integration in the whole B method and its previous notorious applications in the railway domain [45], with a supporting community and a company (ClearSy[17]) that offers support to developers at various levels.

RQ2: What issues should be considered when using Atelier B? When planning to integrate Atelier B in the development process, afirst issue to consider is its strong mathematical foundation in theorem-proving techniques (**theorem-proving background**). Hence, the cost of training developers not familiar with such techniques is non-negligible and must be taken into account. One of the two assessors disagrees on this point: "Usage of (interactive proof) in Atelier B only requires very basic knowledge on what is a proof and how to conduct a correct proof. Any hard scientist or engineer should be able to use the interactive prover to discharge most proof obligations with little training (one or two days)". Contrary to the case of model checking, it is **not a fully automatic** verification technique; it requires more human effort and expertise. Moreover, Atelier B is not well suited when properties related to the temporal evolution of a system are to be analyzed, e.g., through temporal logic (**no temporal properties**). Indeed, in this case a model checker like ProB may be preferable. Finally, Atelier B inherits from Event B some difficulties in modeling concurrent systems and interacting components (**limited parallelism**). However, one assessor mentions it is possible to overcome this problem: "The tool provides support for [. . .] a variant of Event B with a more versatile input language. We have conducted formal analysis of distributed (railways) systems with this language using Atelier B".

RQ1: When to use tools such as ProB? The main characteristic of ProB is that it allows the user to observe, simulate, and model check the dynamic behavior of an Event B state machine. It mightfit well the initial needs of observing the behavior of a prototypical design (**R**), as well as the need of proving dynamic properties of the possible evolutions of a more established but high-level design (**H**). The strong points of ProB are its capability of verifying properties expressed both as linear- or branching-time formulas, the possibility to reason on both state and event properties, the availability of multiple model-checking strategies (**FV-MC**), and the possibility of verifying the consistency of an implementation with respect to its abstract specification in terms of trace refinement (**FV-RC**).

[17]https://www.clearsy.com

ProB is in general a veryfl exible tool, which allows exporting the developed design to other tools of the Event B ecosystem, like Atelier B. Moreover, ProB allows model checking specifications imported by other frameworks such as SPIN or FDR4, providing additional verification capabilities with respect to those provided by the respective tools. This feature, together with the multiple verification techniques supported, makes the tool suitable for users who are skilled in multiple formal methods (**Multiple FM**), but also for novices who wish to explore different techniques.

RQ2: What issues should be considered when using ProB? As for Atelier B, an issue to consider is the limited parallelism (**limited parallelism**), although the assessors disagree with this point. Furthermore, one of them also observed: "ProB is a verification tool (similar to SPIN) and not a modeling tool (like Atelier B); it needs to work with a complementary modeling tool (like Atelier B or Rodin)" (**not design oriented**).

4.5 Verification-Focused Engines

SPIN and NuSMV are two verification frameworks that have a long history of use[18]. Their main common characteristic is their orientation towards large-scale verification through model checking, and the availability of multiple options to tackle the state-space explosion problem [56].

RQ1: When to use tools such as NuSMV or SPIN? These frameworks are not directly inspired by a specific abstract design methodology, but rather focus on the *efficient* model checking of models encoded in their own specific modeling language. To contrast the problem of state-space explosion, SPIN relies on an on-thefl y (distributed) model-checking approach, while NuSMV (and its latest evolution nuXmv) efficiently exploits symbolic (BDD-based) and SMT-based model-checking approaches (**FV-MC**).

A SPIN or NuSMV specification can hardly be seen as a friendly notation for the unambiguous sharing of a design among different stakeholders in the requirements phase. Nevertheless, sometimes a translation of a system design into Promela (the specification language of SPIN) or NuSMV can be an effective way to verify properties of the initial high-level design (**H**). The main difference between the languages of SPIN and NuSMV is that thefi rst one is oriented towards the design of architectures based on concurrent asynchronous processes communicating through message passing, while the second one is more oriented towards datafl ow synchronous architectures. Writing in Promela is close to **C programming**, which makes SPIN suitable for detailed designs (**D**). According to one of the experts, users of NuSMV benefit from knowing "the theory of **symbolic model checking**, e.g., to understand how the variable ordering effects efficiency". Finally, while SPIN is limited to the verification of linear-time properties (LTL), NuSMV allows the verification of both linear- and branching-time properties (CTL).

RQ2: What issues should be considered when using NuSMV or SPIN? When the system under investigation becomes rather complex, the effective use of the tools requires a deep experienced knowledge of the verification framework (**confidence with framework**). The choice of the appropriate execution options may become essential

[18]cf. https://bit.ly/2Wj7ocg and https://bit.ly/2BdYlAz

for the success of the verification task. Moreover, even if the underlying techniques on which the tools are based are able to reduce the impact of the state-space explosion problems, surely they cannot be considered a silver bullet to solve them. Theorem proving or compositional verification are alternative techniques that might produce better results when SPIN or NuSMV fail their verification. Finally, the fact that NuSMV takes only infinite paths into consideration during verification—allfi nite paths are simply ignored—might create difficulties in correctly encoding and verifying software designs (**specification language difficult to master**). It is true that one can add self-loops tofi nal states and solve the problem in many cases. However, it is not easy to add a self-loop tofi nal states if you do not statically know which states arefi nal, e.g., because of the possible presence of some unknown, complex to generate, deadlocks. Given the focus on verification, to the quality of the user interface is taken in minor consideration in SPIN (**limited GUI**), and not considered at all by NuSMV (**no GUI**). However, the SPIN assessor says: "there is a quite simple GUI, but the set of command line switches is large, so I consider the UI really not minimal".

4.6 Tools for Compositional Analysis

Process algebras [35] are formally defined approaches for modeling concurrent systems, and are equipped with rigorous theories of equivalence of behaviors that can be applied to support compositional analysis. Relevant examples of process algebras are LOTOS, at the root of the CADP framework, and CSP_M, at the root of FDR4.

RQ1: When to use tools such as CADP or FDR4? These tools rely on a specification language with a formally defined (tool-independent) semantics. This allows the definition of specifications that can be shared among the various stakeholders, being certain of their unambiguous meaning. Hence, the software development phases in which these tools can be applied at best are those at the early stages of the life cycle, like formalization and verification of requirements or high-level system design (**R/H**). In the case of CADP, the framework provides some support also for testing and for the detailed design and actual program code generation (**D/I**). CADP is also equipped with facilities for abstracting, minimizing, and proving equivalence of behaviors (**FV-EC**). While CADP allows the specification of properties using the μ-calculus (a powerful temporal logic subsuming both LTL and CTL), FDR4 adopts several refinement relations for ensuring the correctness of a model with respect to a higher level abstract specification. As observed by the assessor: "CADP provides tools for Interactive Markov Chains, Discrete- and Continuous-Time Markov Chains", making it suitable for quantitative analysis (**Q**). The assessor also pointed to the recent addition of a probabilistic model checker to the CADP toolbox.

In our specific case, the reference model is composed of just three components. However, one might be interested in checking the behavior of richer systems (e.g., with more OBUs and more interacting RBCs). In this case, a compositional approach that allows the verification of an asynchronous large-scale system-of-systems (**SoS**) by composing it in parallel with a (minimized) abstraction of all the other components would be very useful to avoid the problems of state-space explosion usually arising when one has a system with many concurrent objects under scrutiny.

RQ2: What issues should be considered when using CADP or FDR4? System specifications need to be provided in textual format for these tools, which may appear less intuitive or user friendly than a drawing, especially when the tool has a quite minimal user interface as in case of CADP (**limited GUI**). Initially the assessor disagreed with this observation, but after some interaction he said: "few graphical tools [exist] (simulators, Eucalyptus) which are functional but a bit old-fashioned in their look-and-feel". Concerning issues with CADP the assessor also observed: "CADP is a toolbox, which can be used for many different purposes. As a counterpart of this richness andfl exibility, beginners are disoriented, because they do not know where to start and which tools to select" (**toolbox**). Finally, FDR4 is not well suited when properties related to the temporal evolution of a system have to be analyzed (**no temporal properties**).

5 DISCUSSION

From the judgment study, we can summarize the following main take-away messages about the different tools:

- Simulink is appropriate for both early prototyping and detailed design towards code generation;
- UMC is appropriate for initial prototyping, when one wants to adopt a design based on UML state machines to facilitate communication with different stakeholders, but also wants formal verification capabilities;
- UPPAAL SMC is appropriate when one needs to focus on the verification of quantitative properties involving probabilistic and real-time aspects;
- NuSMV and SPIN are appropriate when the system, or composition of systems, has a rather large state space and one wants to verify temporal logic properties;
- Atelier B and ProB are the right choice for top-down development of mainly monolithic systems, with complementary verification capabilities: Atelier B supports theorem proving of invariants, while ProB supports model checking;
- CADP and FDR4 are appropriate when a clear algebraic specification is desired, and when the system under design has the structure of a concurrent set of asynchronous communicating entities (systems-of-systems).

Below, we discuss the implications of this study distinguishing the viewpoints of practitioners and researchers.

5.1 Implications for Practice

Formal methods diversity. Based on the observation on the primary strengths of each tool, we argue that, to address all the design-related needs of the railway process (e.g., qualitative and quantitative verification, simulation, low-/high-level design), a combination of methods and tools would be required. Guided by the outcomes of the current research, and sill within ASTRail, the authors combined Simulink for prototyping, UPPAAL for verification of quantitative properties and ProB for model checking qualitative ones [9, 31]. One of the main issues encountered was the absence of *interoperability* of the tools, which led to the need to manually translate the initial models, with the obvious issues of consistency. In practice, a company may not want to rely on multiple tools that, although somewhat complementary, are not integrated within the same framework. In this case, it is advisable to identify which aspects of the development are more relevant (e.g., early prototyping of a system-of-systems vs. detailed design of a single system), and select the most appropriate tool to address them.

Expert diversity. To fully tackle the formal development of railway systems, not only a combination of tools is required but also the involvement of different experts—in the domain, in the design, and in multiple types of formal verification techniques (see the diversity of *facilitating user skills* and background *issues* in Table 2). These aspects should be taken into account by companies willing to introduce formal methods in their process. In practice, it may be hard for a company to have different tool experts at hand. The effort needed to train engineers in formal methods is not negligible as the subject is undeniably hard, and high expertise with a tool is required to handle industrial problems. Therefore, a feasible approach today for railway companies is to rely on external consultants with an academic background on formal methods, and invest in communication between railway experts and formal method experts, possibly allowing them to share the same work space.

Relevance for the railway industry. The conclusion of this research could appear straightforward to formal methods academics. However, it should be noticed that this systematic study on formal tools was required by the EU railway industry itself, as the Shift2Rail initiative funding the ASTRail project involves the main stakeholders in this domain. Although formal methods are highly recommended by the CENELEC norms for the development of high-integrity railway systems [25], the knowledge of formal methods, especially at management level, is limited, and it is thus necessary to make explicit for this specific public what existing technologies can and cannot do for them. We argue that this study is a step forward in this direction. Further steps, currently undertaken by the authors within the 4SECURail project[19], concern the cost-benefit analysis of introducing formal methods in railway companies.

5.2 Implications for Research

Tool improvements: usability, learnability, maturity. While several formal tools are available to be used for railway development, some issues emerge when considering their real-world application, such as the absence of constructs to model parallelism, limitations in terms of hierarchical modeling, etc. Researchers working on specific tools are called to refer to Table 2 and address the issues raised in future releases of the tools. Many common issues are related to usability, learnability and maturity aspects (e.g., limited GUI, specific technical background, maturity of the theory, prototypical tool). Interestingly, these are also considered among the primary needs by railway stakeholders, according to a recent survey [8]. As most of the tools originate from academic contexts, they are often used as testbed to demonstrate novel techniques. However, the more a technique is advanced, the more its implementation is prototypical, and industry-relevant aspects tend to be neglected. A development model for tool providers could be to maintain two versions of their tool: a research-oriented one, implementing the recent techniques, and a practice-oriented one, with a focus on usability, learnability, and maturity.

[19]www.4securail.eu

Relevance of the research method and human aspects of formal design. This paper presents thefi rst design of a rigorous judgment study in formal methods. We believe that the systematic approach followed can be considered by other researchers dealing with similar contexts, in which a comparison between tools or methods is required, but other research methods (e.g., controlled experiments, surveys) are not applicable. This can happen if (1) the topic is complex and only experts can participate as human subjects of the study; (2) a hands-on experience is needed to trigger experts' reflections; (3) a limited number of experts is available for each method or tool; and (4) the nature of the inquiry is exploratory, with a broad scope.

In principle, the tools' documentation could have been used by the designers to elicit most of the information presented. However, without a hands-on experience like the one introduced in our method, the risk is to ignore which are the most *relevant* aspects for the railway context. The adopted empirical approach enabled the designers to reflect on those aspects (concerning phase, type of system, etc.) that are most characteristic with respect to the other tools. This provides a grounded and focused synthesis that could hardly be achieved by looking solely at the documentation. Furthermore, thanks to the adopted research design, and in particular the introduction of assessors, we also noticed that some tools' aspects that may be seen as objective (e.g., limited parallelism, Sect. 4.4, see issues in *italic* in Table 2) appeared debatable, and different experts gave different viewpoints. This suggests that the *subjectivity* of the user plays a crucial role also in formal methods, and calls for more empirical studies focused on human aspects of formal design.

6 CONCLUSION

This paper presents a qualitative evaluation of 9 formal tools in the context of railway systems design by means of a judgment study. The study involved 17 experts in formal methods applied to railways, and produced an evaluation and comparison table (see Table 2), in which the tools are characterized by their suitability for a certain development context, and by the issues that users should consider when adopting a formal tool. The paper makes an effort to provide indications to companies interested in adopting formal methods, and to make the information from formal methods experts accessible also to a broader software engineering audience. The produced categorization can also be used as a framework for comparing other formal tools, and offers a baseline for further, more fine-grained analysis of the characteristics of the tools. As future work, we will complement the current analysis with a systematic DESMET [44] evaluation over technical features, and a usability study. This will produce a comprehensive view of the tools' characteristics, which can be referred also by tools' developers, and can be updated as new features or knowledge become available.

ACKNOWLEDGMENTS

This work has been partially funded by the ASTRail and 4SECURail projects. These projects received funding from the Shift2Rail Joint Undertaking under the European Union's Horizon 2020 research and innovation programme under grant agreements 777561 and 881775. 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.

REFERENCES

[1] Jean-Raymond Abrial. 2007. Formal Methods: Theory Becoming Practice. *Journal of Universal Computer Science* 13, 5 (2007), 619–628. https://doi.org/10.3217/jucs-013-05-0619

[2] Jean-Raymond Abrial and Jean-Raymond Abrial. 2005. *The B-book: assigning programs to meanings.* Cambridge University Press.

[3] Paolo Arcaini, Pavel Ježek, and Jan Kofroň. 2018. Modelling the Hybrid ERTMS/ETCS Level 3 Case Study in Spin. In *Proceedings of the 6th International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z (ABZ 2018) (Lecture Notes in Computer Science)*, Michael Butler, Alexander Raschke, Thai Son Hoang, and Klaus Reichl (Eds.), Vol. 10817. Springer, Germany, 277–291. https://doi.org/10.1007/978-3-319-91271-4_19

[4] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking.* MIT Press.

[5] Maarten Bartholomeus, Bas Luttik, and Tim Willemse. 2018. Modeling and Analysing ERTMS Hybrid Level 3 with the mCRL2 Toolset. In *Proceedings of the 23rd International Conference on Formal Methods for Industrial Critical Systems (FMICS 2018) (Lecture Notes in Computer Science)*, Falk Howar and Jiří Barnat (Eds.), Vol. 11119. Springer, Germany, 98–114. https://doi.org/10.1007/978-3-030-00244-2_7

[6] Davide Basile, Felicita Di Giandomenico, and Stefania Gnesi. 2017. Statistical Model Checking of an Energy-Saving Cyber-Physical System in the Railway Domain. In *Proceedings of the 32nd Symposium on Applied Computing (SAC 2017).* ACM, USA, 1356–1363. https://doi.org/10.1145/3019612.3019824

[7] Davide Basile, Maurice H. ter Beek, and Vincenzo Ciancia. 2018. Statistical Model Checking of a Moving Block Railway Signalling Scenario with Uppaal SMC. In *Proceedings of the 8th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation — Verification (ISoLA 2018) (Lecture Notes in Computer Science)*, Tiziana Margaria and Bernhard Steffen (Eds.), Vol. 11245. Springer, Germany, 372–391. https://doi.org/10.1007/978-3-030-03421-4_24

[8] Davide Basile, Maurice H. ter Beek, Alessandro Fantechi, Stefania Gnesi, Franco Mazzanti, Andrea Piattino, Daniele Trentini, and Alessio Ferrari. 2018. On the Industrial Uptake of Formal Methods in the Railway Domain – A Survey with Stakeholders. In *Proceedings of the 14th International Conference on Integrated Formal Methods (iFM 2018) (Lecture Notes in Computer Science)*, Carlo A. Furia and Kirsten Winter (Eds.), Vol. 11023. Springer, Germany, 20–29. https://doi.org/10.1007/978-3-319-98938-9_2

[9] Davide Basile, Maurice H. ter Beek, Alessio Ferrari, and Axel Legay. 2019. Modelling and Analysing ERTMS L3 Moving Block Railway Signalling with Simulink and UPPAAL SMC. In *Proceedings of the 24th International Conference on Formal Methods for Industrial Critical Systems (FMICS 2019) (Lecture Notes in Computer Science)*, Kim G. Larsen and Tim Willemse (Eds.), Vol. 11687. Springer, Germany. https://doi.org/10.1007/978-3-030-27008-7_1

[10] Maurice H. ter Beek, Arne Borälv, Alessandro Fantechi, Alessio Ferrari, Stefania Gnesi, Christer Löfving, and Franco Mazzanti. 2019. Adopting Formal Methods in an Industrial Setting: The Railways Case. In *Formal Methods – The Next 30 Years — Proceedings of the 3rd World Congress on Formal Methods (FM'19) (Lecture Notes in Computer Science)*, Maurice H. ter Beek, Annabelle McIver, and José N. Oliveira (Eds.), Vol. 11800. Springer, Germany, 762–772. https://doi.org/10.1007/978-3-030-30942-8_46

[11] Maurice H. ter Beek, Stefania Gnesi, and Alexander Knapp. 2018. Formal methods for transport systems. *International Journal on Software Tools for Technology Transfer* 20, 3 (2018), 237–241. https://doi.org/10.1007/s10009-018-0487-4

[12] Ulrich Berger, Phillip James, Andrew Lawrence, Markus Roggenbach, and Monika Seisenberger. 2018. Verification of the European Rail Traffic Management System in Real-Time Maude. *Science of Computer Programming* 154 (2018), 61–88. https://doi.org/10.1016/j.scico.2017.10.011

[13] Yves Bertot and Pierre Castéran. 2004. *Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions.* Springer. https://doi.org/10.1007/978-3-662-07964-5

[14] Wolfgang Bibel. 2013. *Automated theorem proving.* Springer Science & Business Media.

[15] Dines Bjørner. 2003. New Results and Trends in Formal Techniques and Tools for the Development of Software for Transportation Systems — A Review. In *Proceedings of the 4th Symposium on Formal Methods for Railway Operation and Control Systems (FORMS 2003)*, Géza Tarnai and Eckehard Schnieder (Eds.). L'Harmattan, Hungary.

[16] Mark Bosschaart, Egidio Quaglietta, Bob Janssen, and Rob M. P. Goverde. 2015. Efficient formalization of railway interlocking data in RailML. *Information Systems* 49 (2015), 126–141. https://doi.org/10.1016/j.is.2014.11.007

[17] Jean-Louis Boulanger (Ed.). 2014. *Formal Methods Applied to Industrial Complex Systems — Implementation of the B Method.* John Wiley & Sons, USA. https://doi.org/10.1002/9781119002727

[18] Quentin Cappart, Christophe Limbrée, Pierre Schaus, Jean Quilbeuf, Louis-Marie Traonouez, and Axel Legay. 2017. Verification of Interlocking Systems Using Statistical Model Checking. In *Proceedings of the 18th International Symposium on High Assurance Systems Engineering (HASE 2017).* IEEE, 61–68. https://doi.org/10.1109/HASE.2017.10

[19] Angelo Chiappini, Alessandro Cimatti, Luca Macchi, Oscar Rebollo, Marco Roveri, Angelo Susi, Stefano Tonetta, and Berardino Vittorini. 2010. Formalization and Validation of a subset of the European Train Control System. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE 2010)*, Vol. 2. ACM, USA, 109–118. https://doi.org/10.1145/1810295.1810312

[20] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. 1999. *Model Checking.* MIT Press.

[21] Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem (Eds.). 2018. *Handbook of Model Checking.* Springer. https://doi.org/10.1007/978-3-319-10575-8

[22] Nicolas Coste, Hubert Garavel, Holger Hermanns, Frédéric Lang, Radu Mateescu, and Wendelin Serwe. 2010. Ten Years of Performance Evaluation for Concurrent Systems Using CADP. In *Proceedings of the 4th International Symposium on Leveraging Applications of Formal Methods, Verification, and Validation (ISoLA 2010) (Lecture Notes in Computer Science)*, Tiziana Margaria and Bernhard Steffen (Eds.), Vol. 6416. Springer, 128–142. https://doi.org/10.1007/978-3-642-16561-0_18

[23] Alcino Cunha and Nuno Macedo. 2018. Validating the Hybrid ERTMS/ETCS Level 3 Concept with Electrum. In *Proceedings of the 6th International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z (ABZ 2018) (Lecture Notes in Computer Science)*, Michael Butler, Alexander Raschke, Thai Son Hoang, and Klaus Reichl (Eds.), Vol. 10817. Springer, Germany, 307–321. https://doi.org/10.1007/978-3-319-91271-4_21

[24] James B Dabney and Thomas L Harman. 2004. *Mastering simulink.* Pearson.

[25] European Committee for Electrotechnical Standardization. 2011. CENELEC EN 50128 — Railway applications – Communication, signalling and processing systems – Software for railway control and protection systems. (1 June 2011). https://standards.globalspec.com/std/1678027/cenelec-en-50128.

[26] Alessandro Fantechi. 2013. Twenty-Five Years of Formal Methods and Railways: What Next?. In *Software Engineering and Formal Methods — Revised Selected Papers of the SEFM 2013 Collocated Workshops: BEAT2, WS-FMDS, FM-RAIL-Bok, MoKMaSD, and OpenCert (Lecture Notes in Computer Science)*, Steve Counsell and Manuel Núñez (Eds.), Vol. 8368. Springer, Germany, 167–183. https://doi.org/10.1007/978-3-319-05032-4_13

[27] Alessandro Fantechi, Wan Fokkink, and Angelo Morzenti. 2013. Some Trends in Formal Methods Applications to Railway Signaling. In *Formal Methods for Industrial Critical Systems: A Survey of Applications*, Stefania Gnesi and Tiziana Margaria (Eds.). John Wiley & Sons, USA, Chapter 4, 61–84. https://doi.org/10.1002/9781118459898.ch4

[28] Alessio Ferrari, Alessandro Fantechi, Stefania Gnesi, and Gianluca Magnani. 2013. Model-based development and formal methods in the railway industry. *IEEE software* 30, 3 (2013), 28–34. https://doi.org/10.1109/MS.2013.44

[29] Alessio Ferrari, Daniele Grasso, Gianluca Magnani, Alessandro Fantechi, and Matteo Tempestini. 2013. The Metrô Rio case study. *Science of Computer Programming* 78, 7 (2013), 828–842. https://doi.org/10.1016/j.scico.2012.04.003

[30] Alessio Ferrari, Franco Mazzanti, and Davide Basile. 2019. Moving-block System: Requirements and Formal Models. (Aug. 2019). https://doi.org/10.5281/zenodo.3375494

[31] Alessio Ferrari, Franco Mazzanti, and Andrea Piattino. 2019. ASTRail Deliverable 4.3, Task 4.4 - Supplementary Material. (Aug. 2019). https://doi.org/10.5281/zenodo.3377823

[32] Alessio Ferrari, Maurice H. ter Beek, Franco Mazzanti, Davide Basile, Alessandro Fantechi, Stefania Gnesi, Andrea Piattino, and Daniele Trentini. 2019. Survey on Formal Methods and Tools in Railways: The ASTRail Approach. In *Proceedings of the 3rd International Conference on Reliability, Safety, and Security of Railway Systems — Modelling, Analysis, Verification, and Certification (RSSRail'19) (Lecture Notes in Computer Science)*, Simon Collart-Dutilleul, Thierry Lecomte, and Alexander Romanovsky (Eds.), Vol. 11495. Springer, Germany, 226–241. https://doi.org/10.1007/978-3-030-18744-6_15

[33] Melvin Fitting. 2012. *First-order logic and automated theorem proving.* Springer Science & Business Media.

[34] Francesco Flammini (Ed.). 2012. *Railway Safety, Reliability, and Security: Technologies and Systems Engineering.* IGI Global, USA. https://doi.org/10.4018/978-1-4666-1643-1

[35] Wan Fokkink. 2000. *Introduction to Process Algebra.* Springer. https://doi.org/10.1007/978-3-662-04293-9

[36] Hubert Garavel, Frédéric Lang, Radu Mateescu, and Wendelin Serwe. 2013. CADP 2011: a toolbox for the construction and analysis of distributed processes. *International Journal on Software Tools for Technology Transfer* 15, 2 (2013), 89–107. https://doi.org/10.1007/s10009-012-0244-z

[37] Mohamed Ghazel. 2014. Formalizing a subset of ERTMS/ETCS specifications for verification purposes. *Transportation Research Part C: Emerging Technologies* 42 (2014), 60–75. https://doi.org/10.1016/j.trc.2014.02.002

[38] Thomas Gibson-Robinson, Philip J. Armstrong, Alexandre Boulgakov, and A. William Roscoe. 2016. FDR3: a parallel refinement checker for CSP. *International Journal on Software Tools for Technology Transfer* 18, 2, 149–167. https://doi.org/10.1007/s10009-015-0377-y

[39] Anne E. Haxthausen, Hoang Nga Nguyen, and Markus Roggenbach. 2016. Comparing Formal Verification Approaches of Interlocking Systems. In *Proceedings*

of the 1st International Conference on Reliability, Safety, and Security of Railway Systems: Modelling, Analysis, Verification, and Certification (Lecture Notes in Computer Science), Thierry Lecomte, Ralf Pinger, and Alexander B. Romanovsky (Eds.), Vol. 9707. Springer, Germany, 160–177. https://doi.org/10.1007/978-3-319-33951-1_12

[40] Anne E. Haxthausen, Jan Peleska, and Sebastian Kinder. 2011. A formal approach for the construction and verification of railway control systems. *Formal Aspects of Computing* 23, 2 (2011), 191–219. https://doi.org/10.1007/s00165-009-0143-6

[41] Gerard J Holzmann. 2004. *The SPIN model checker: Primer and reference manual*. Vol. 1003. Addison-Wesley Reading.

[42] Alexei Iliasov, Dominic Taylor, Linas Laibinis, and Alexander B. Romanovsky. 2018. Formal Verification of Signalling Programs with SafeCap. In *Proceedings of the 37th International Conference on Computer Safety, Reliability, and Security (SAFECOMP 2018) (Lecture Notes in Computer Science)*, Barbara Gallina, Amund Skavhaug, and Friedemann Bitsch (Eds.), Vol. 11093. Springer, Germany, 91–106. https://doi.org/10.1007/978-3-319-99130-6_7

[43] Phillip James, Faron Moller, Hoang Nga Nguyen, Markus Roggenbach, Steve Schneider, and Helen Treharne. 2014. Techniques for modelling and verifying railway interlockings. *International Journal on Software Tools for Technology Transfer* 16 (2014), 685–711. https://doi.org/10.1007/s10009-014-0304-7

[44] Barbara Kitchenham, Stephen Linkman, and David Law. 1997. DESMET: a methodology for evaluating software engineering methods and tools. *Computing & Control Engineering Journal* 8, 3 (1997), 120–126. https://doi.org/10.1049/cce:19970304

[45] Thierry Lecomte, David Déharbe, Étienne Prun, and Erwan Mottin. 2017. Applying a Formal Method in Industry: A 25-Year Trajectory. In *Proceedings of the 20th Brazilian Symposium on Formal Methods: Foundations and Applications (SBMF 2017) (Lecture Notes in Computer Science)*, Simone Cavalheiro and José Fiadeiro (Eds.), Vol. 10623. Springer, Germany, 70–87. https://doi.org/10.1007/978-3-319-70848-5_6

[46] Axel Legay, Benoît Delahaye, and Saddek Bensalem. 2010. Statistical Model Checking: An overview. In *Proceedings of the 1st International Conference on Runtime Verification (RV 2010) (Lecture Notes in Computer Science)*, Howard Barringer, Ylies Falcone, Bernd Finkbeiner, Klaus Havelund, Insup Lee, Gordon Pace, Grigore Roşu, Oleg Sokolsky, and Nikolai Tillmann (Eds.), Vol. 6418. Springer, 122–135. https://doi.org/10.1007/978-3-642-16612-9_11

[47] Lawrence Leung. 2015. Validity, reliability, and generalizability in qualitative research. *Journal of family medicine and primary care* 4, 3 (2015), 324. https://doi.org/10.4103/2249-4863.161306

[48] Michael Leuschel and Michael J. Butler. 2003. ProB: A Model Checker for B. In *Proceedings of the 12th International Symposium on Formal Methods (FME 2003) (Lecture Notes in Computer Science)*, Keijiro Araki, Stefania Gnesi, and Dino Mandrioli (Eds.), Vol. 2805. Springer, 855–874. https://doi.org/0.1007/978-3-540-45236-2_46

[49] Michael Leuschel, Jérôme Falampin, Fabian Fritz, and Daniel Plagge. 2011. Automated property verification for large scale B models with ProB. *Formal Aspects of Computing* 23, 6 (2011), 683–709. https://doi.org/10.1007/s00165-010-0172-1

[50] Amel Mammar, Marc Frappier, Steve J. Tueno Fotso, and Régine Laleau. 2018. An Event-B Model of the Hybrid ERTMS/ETCS Level 3 Standard. In *Proceedings of the 6th International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z (ABZ 2018) (Lecture Notes in Computer Science)*, Michael Butler, Alexander Raschke, Thai Son Hoang, and Klaus Reichl (Eds.), Vol. 10817. Springer, Germany, 353–366. https://doi.org/10.1007/978-3-319-91271-4_24

[51] Franco Mazzanti and Alessio Ferrari. 2018. Ten Diverse Formal Models for a CBTC Automatic Train Supervision System. In *Proceedings of the 3rd Workshop on Models for Formal Analysis of Real Systems and the 6th International Workshop on Verification and Program Transformation (MARS/VPT 2018) (Electronic Proceedings in Theoretical Computer Science)*, John P. Gallagher, Rob van Glabbeek, and Wendelin Serwe (Eds.), Vol. 268. 104–149. https://doi.org/10.4204/EPTCS.268.4

[52] Franco Mazzanti, Alessio Ferrari, and Giorgio O. Spagnolo. 2018. Towards formal methods diversity in railways: an experience report with seven frameworks. *International Journal on Software Tools for Technology Transfer* 20, 3 (2018), 263–288. https://doi.org/10.1007/s10009-018-0488-3

[53] Faron Moller, Hoang Nga Nguyen, Markus Roggenbach, Steve Schneider, and Helen Treharne. 2013. Defining and Model Checking Abstractions of Complex Railway Models Using CSP||B. In *Hardware and Software: Verification and Testing — Revised Selected Papers of the 8th International Haifa Verification Conference (HVC 2012) (Lecture Notes in Computer Science)*, Armin Biere, Amir Nahir, and Tanja Vos (Eds.), Vol. 7857. Springer, Germany, 193–208. https://doi.org/10.1007/978-3-642-39611-3_20

[54] Roberto Nardone, Ugo Gentile, Massimo Benerecetti, Adriano Peron, Valeria Vittorini, Stefano Marrone, and Nicola Mazzocca. 2016. Modeling Railway Control Systems in Promela. In *Formal Techniques for Safety-Critical Systems — Revised Selected Papers of the 4th International Workshop on Formal Techniques for Safety-Critical Systems (FTSCS 2015) (Communications in Computer and Information Science)*, Cyrille Artho and Peter C. Ölveczky (Eds.), Vol. 596. Springer, Germany, 121–136. https://doi.org/10.1007/978-3-319-29510-7_7

[55] Object Management Group (OMG). 2017. Unified Modeling Language, Version 2.5.1. https://www.omg.org/spec/UML/About-UML/. (2017).

[56] Radek Pelánek. 2009. Fighting State Space Explosion: Review and Evaluation. In *Proceedings of the 13th International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2008) (Lecture Notes in Computer Science)*, Darren Cofer and Alessandro Fantechi (Eds.), Vol. 5596. Springer, 37–52. https://doi.org/10.1007/978-3-642-03240-0_7

[57] Klaas-Jan Stol and Brian Fitzgerald. 2018. The ABC of software engineering research. *ACM Transactions on Software Engineering and Methodology* 27, 3 (2018), 11. https://doi.org/10.1145/3241743

[58] Mojtaba Vaismoradi, Hannele Turunen, and Terese Bondas. 2013. Content analysis and thematic analysis: Implications for conducting a qualitative descriptive study. *Nursing & health sciences* 15, 3 (2013), 398–405. https://doi.org/10.1111/nhs.12048

[59] Somsak Vanit-Anunchai. 2018. Modelling and simulating a Thai railway signalling system using Coloured Petri Nets. *International Journal on Software Tools for Technology Transfer* 20, 3 (2018), 243–262. https://doi.org/10.1007/s10009-018-0482-9

[60] Linh Hong Vu, Anne E. Haxthausen, and Jan Peleska. 2017. Formal modelling and verification of interlocking systems featuring sequential release. *Science of Computer Programming* 133 (2017), 91–115. https://doi.org/10.1016/j.scico.2016.05.010

[61] Jeannette M. Wing. 1990. A specifier's introduction to formal methods. *IEEE Computer* 23, 9 (1990), 8–22. https://doi.org/10.1109/2.58215

[62] Kirsten Winter and Neil J. Robinson. 2003. Modelling Large Railway Interlockings and Model Checking Small Ones. In *Proceedings of the 26th Australasian Computer Science Conference (ACSC 2003) (Conferences in Research and Practice in Information Technology)*, Michael J. Oudshoorn (Ed.), Vol. 16. Australian Computer Society, Australia, 309–316. http://crpit.com/confpapers/CRPITV16Winter.pdf.

[63] Jim Woodcock, Peter G. Larsen, Juan Bicarregui, and John Fitzgerald. 2009. Formal methods: Practice and experience. *Comput. Surveys* 41, 4 (2009), 19:1–19:36. https://doi.org/10.1145/1592434.1592436