12 A New Formal Methods Guidebook for the Railway Signalling Domain

Daniel Schwencke; DLR, Institute of Transportation Systems, Braunschweig, Germany
Arne Borälv; Prover Technologies, Stockholm, Sweden
Luis-Fernando Mejia; Alstom, Paris, France

12.1 Motivation for the Guidebook

Considerable know-how about formals methods (FMs) exists in the railway signalling domain [1, 2], and FMs have been successfully applied e.g. for verification of interlockings [3] or development of computer-based train control systems [4]. Some railway infrastructure managers, such as RATP, New York City Transit, Stockholm Metro and Trafikverket even prescribe formal safety verification for some types of relay-based or computerised interlocking systems. But FMs expertise is not generally available or widespread, and apart from a general recommendation in [5], there is a lack of FMs integration into standards, of recommendations for FMs use and of guidance on where and how to employ them. For these reasons, and due to interest from Europe's Rails System Pillar, TD2.7 of Shift2Rail is currently preparing a FMs guidebook as part of the work in WP10 of the X2Rail-5 project. This guidebook expands on X2Rail-2 work [6] and aims to document know-how, experience, and recommendations, to pave the way for wider use of FMs.

The scope of the guidebook includes today's and future railway signalling systems; its view on FMs is geared towards typical properties and conditions of these such as high RAMS[1] requirements and high configurability. The target audience includes infrastructure managers, suppliers, railway engineers, railway signalling initiatives and projects.

In the following, the guidebook contents are briefly presented, including why, when and for what purpose to apply FMs, what FMs are and which steps their application follows. Afterwards, an example application of the guidebook's FMs concepts is described, based on WP10's FMs demonstrator for ETCS[2] Level 3 with moving block. Finally, a summary, conclusions from the guidebook creation, and an outlook on the finalisation of the guidebook are given.

12.1.1 Reasons to Apply FMs (Now)

Current developments like upcoming new functionality (ATO[2], moving block, train integrity, new train positioning approaches) and new modular standard architectures (EULYNX [7], RCA[2]

[1]Abbreviations: RAMS = Reliability, Availability, Maintainability and Safety; ETCS = European Train Control System; ATO = Automated Train Operation; RCA = Reference CCS (Control Command and Signalling) Architecture; SIL = Safety Integrity Level; V&V = Verification and Validation.

[8], and their continuation in Europe's Rails System Pillar) constitute both a request for the capabilities provided by FMs and a unique opportunity to introduce FMs on larger scale. In this situation of growing system and especially software complexity, FMs can be a valuable, meanwhile mature tool to maintain safety and to limit time and costs spent on a signalling system during its life cycle. The main gains from FMs use are presented in Table 12-1.

Table 12-1: Benefits from FMs w.r.t. current challenges in railway signaling

Aspect	**Current situation**	**How to benefit from FMs**
Time-to-market and predictable schedules	Long and unpredictable schedules, systems costly to procure, develop and maintain	Improve quality of system requirements and tenders using FMs, to find issues earlier, reduce complexity, enable reuse and standardise
High RAMS demands	Traditional methods for specification, architecture, design, implementation and verification of SIL[2] 4 software prevalent although laborious and higher risk of residual errors	Raise trust in and quality and verifiability of implementations, due to *formal verification*[2] of critical system properties (e.g. safety, interoperability), automation of tedious V&V[2] tasks, and valuable feedback, insight and helps to detect and correct mistakes
New system principles	ETCS "game-changer" technologies (ATO, moving block, …) are being specified, aiming to become part of future harmonised standards versions	Define principles and perform analysis and V&V of requirements before harmonisation/ standardisation, to ensure clear and verifiable specifications that multiple stakeholders understand and which ensure safety
Modular architectures and standardisation	Infrastructure managers expect future systems to implement new, modular architectures, to increase competition, reduce costs and support the long-term evolution	Perform *formal development* of a reference model that implements principles and requirements apportioned to its components, to validate the new requirements and processes, foster high-quality modular safety cases, and enable different analyses (e.g. impact analysis, safety, degraded modes, configurability)
Knowledge capitalisation	Expert staff is a scarce resource and bottleneck, many tasks are carried out manually based on expert judgement, for which long experience in railways is required	Use FMs to promote knowledge capitalisation, enable reusability, precise impact analysis and more independence from domain expertise. Use expertise to define and maintain principles and requirements that can be reused, and to prepare automated V&V processes, if possible

[2] An explanation of terms in italics is provided in Section „12.3.5 FMs Taxonomy" below.

12.2 Where to Apply FMs

12.2.2 FMs in the System Life Cycle

The guidebook distinguishes four high-level (HL) phases of a system life cycle in which FMs may be used. They correspond to different roles of FMs in a life cycle and serve as structural basis for presenting FMs uses and recommendations. The phases are presented in Table 12-2 including their names, correspondences to phases from the CENELEC standard [9], relevant activities, possible FMs uses and considerations for being FMs friendly in each phase.

Table 12-2: Activities, FMs use and how to be FMs friendly in the four high-level phases defined in the guidebook (reqs = requirements, SRACs = Safety-Related Application Conditions)

Activities	**FMs use**	**Being FMs friendly**
HL1 – Define Standard Principles and Requirements (CENELEC phases 1-4)		
• Gain understanding of user reqs (e.g. by prototyping) • Formulate and refine implementation-independent ontology • Define contracts between concepts or known subsystems • Create and refine a reference design • Specify mandatory principles and reqs	If activities use FMs, properties will be made precise through formalisation and may be formally proved consistent and preserved during system operation. This will result in well-founded standard principles and high-quality reqs, enabling reuse and a competitive market. FMs use in this phase means most likely extra effort, but will pay off in later phases.	• Identify desired (interoperability, safety, reliability, standards compliance) system properties early (in this phase) • Establish provability of properties early (in this phase) • Strive for self-contained reqs
HL2 – Architecture and Design (CENELEC phase 5)		
• Decompose system • Apportion reqs to subsystems • State assumptions on respective subsystem environment explicitly • Define interfaces between subsystems	• Analysis/V&V of system-level reqs, degraded modes and system initialisation when decomposing the system • Validation of operational procedures whether they fulfil assumptions on system environment • Verification of communication protocols	• Apportion reqs and assumptions such that important system-level properties can be formally verified • Provide abstractions of interfaces suited for FMs use
HL3 – Implement a System (CENELEC phases 6-8)		
• Implement system (as configurable generic application) based on reqs, architecture and design	• Define and verify consistency and correctness reqs for configuration data • Verify generic application or instantiated generic application against reqs • Prove generic properties of code (termination, no dead code, …)	• Formally verify safety properties & consistency during development (not just afterwards) • Carefully select a set of system configuration that exercise a suitable level of coverage of the reqs

Activities	FMs use	Being FMs friendly
HL4 – Assess a System (CENELEC phases 9-10)		
• Collect all SRACs established in previous phases • Validate compliance of the system with the reqs • Collect SRACs from failed verifications	• Verify reqs in scope (at least safety reqs) and demonstrate SRAC sufficiency • Apply additional techniques (e.g. proof checking) in case traditional assessment is to be replaced completely • Perform safety assessment independent of previous phases	• Provide appropriate system/code interfaces for verification of reqs • Avoid complex/non-FM-supported code constructs

12.2.3 Example Purposes for Using FMs

System Inception. A new (type of) system needs to be defined. To identify and define the "right" core concepts, their relations and necessary assumptions (e.g. on the system environment) that will represent a common view of the system, a high-level formal model of the system is created and analysed whether it meets expectations, allows for important usage scenarios, and fulfils basic properties. Model and properties may be refined/reused in later phases e.g. as reference by different actors in projects.

Tender Creation & Verification of Implementation(s) against Tender. Tender requirements are provided to suppliers to implement systems that comply with them. Using a process based on FMs can provide clearly defined tender requirements with less room for interpretation, and formal verification can be used to verify compliance of the implementation to (relevant parts of) the tender. FMs use by suppliers is not necessarily required for this use case.

Development of System Implementation(s). A supplier develops system implementations with the help of FMs, using a process for formal development. The latter means that formal specification and formal verification of system requirements / desired implementation properties are applied to dynamic behaviour and/or static data during the conversion of the system specification into an implementation.

Safety Verification of System Implementation(s). A system has been already developed. The purpose of FMs use is to perform formal verification of safety requirements against a (revenue service) implementation model, to identify any deviations. If formal verification of safety requirements has been carried out already during development, this may help (in various ways) to complete this purpose.

Testing Support. Formal system models can be used for test case generation and, if they are executable, for system simulation. Some test case generation tools rely on formal techniques.

12.3 Understanding What FMs Are

12.3.4 General Overview

A FM enables formalising and analysing the static and dynamic characteristic properties of systems using mathematical models. Typically, a FM has three components:

- A graphical or textual notation whose syntax is defined by a grammar and whose semantics is mathematically defined, allowing for formal proofs.
- A methodology for using the notation to create meaningful and relevant models.
- A set of tools for creating and analysing models, including formal proofs.

There are many different FMs. They may differ in their system model paradigm – e.g. state-based, transition-based, (a)synchronous communications-based – and in their underlying mathematical formalism – e.g. (temporal) set theory, (temporal) automata, type theories.

FMs allow the rigorous formalisation of requirements on the system, and on its design or implementation, and the comprehensive verification of their correctness and consistency. In doing so, FMs allow the improvement of both, the quality of the system's requirements and the conformity of the system's implementation to them.

FMs can be used for the specification, design and verification of software and hardware systems. Experience shows [10] that FMs do not solve all problems of system development: When used for specification, where they are most beneficial, they can raise issues, but they do not guarantee the completeness and relevance of the specification (will not fill the gaps).

Also, FMs do not eliminate the intrinsic complexity of systems. Even with FMs, which certainly help to manage or even to reduce complexity, complex systems can remain difficult to model and analyse. Finally, applying FMs is not self-evident. Important factors of successful FMs use can include:

- A process and organisation of development taking FMs into account.
- The availability of staff experienced in applying FMs to systems of similar complexity in a similar context.
- The close collaboration of FMs staff with system engineers.
- The nature of people who willingly and enthusiastically apply FMs.

12.3.5 FMs Taxonomy

The guidebook distinguishes different FMs activities:

Formal specification is used to formalise and analyse the definition of a system: It creates a model of the static and/or dynamic characteristic properties of a system, which allows to better understand the system and/or to verify that the model is consistent.

Formal verification is used to check that a specification, design or implementation of a system complies with requirements on the system. Based on the purpose of the model, this can check that the model is consistent, that certain properties always hold, or that the model is consistent with another (reference) model.

Formal development is used to implement a system based on formal specification and formal verification integrated into the development process. Formal development therefore includes creating a formal implementation model according to definition and implementation constraints.

12.3.6 Typical Techniques Applied as Part of FMs

Divide and conquer is the process of dividing a problem into smaller, simpler subproblems, whose solutions can be composed to solve the original problem. Functional decomposition is one of many examples of this technique.

Property-oriented reasoning is the process of formalising a system by the properties it must satisfy rather than by the ways it satisfies them. State invariants and assume/guarantee properties, guards and pre- and postconditions of actions are examples of properties.

Abstraction is the process of representing a concept of the system in a simplified form, only including what is needed for the modelling purpose. Generalisation is a form of abstraction: it represents different concepts of a system with common properties by a single concept.

In addition, there are **specific tool-supported techniques**, including

- model transformation – e.g. generation of a formal model from a specification model, design model, or from a software implementation,
- model animation/simulation/execution, which may be used to analyse and validate system behaviour,
- proof search and counter example generation (used for formal verification),
- proof checking, for verifying that a given proof indeed is a correct proof,
- debug capabilities, to isolate and understand the cause of a dynamic situation involving system requirements, and
- test case generation, to automate creation of test suites.

12.3.7 Applying FMs

The FMs guidebook describes a generic process for FMs application that aims to be independent of purpose, life cycle phase, notations and tools, or other characteristics. This process encompasses six generic steps (activities), which are shown as chevron arrows in Figure 12-1. Depending on the context, individual steps may vary in abstraction level, system aspects considered, notations and tools used. Effort spent on a step will vary depending on quality/suitability of inputs (box labelled "User needs, …") and the degree of reuse (e.g. the "Define ontology" step may just refine or extend a pre-existing ontology).

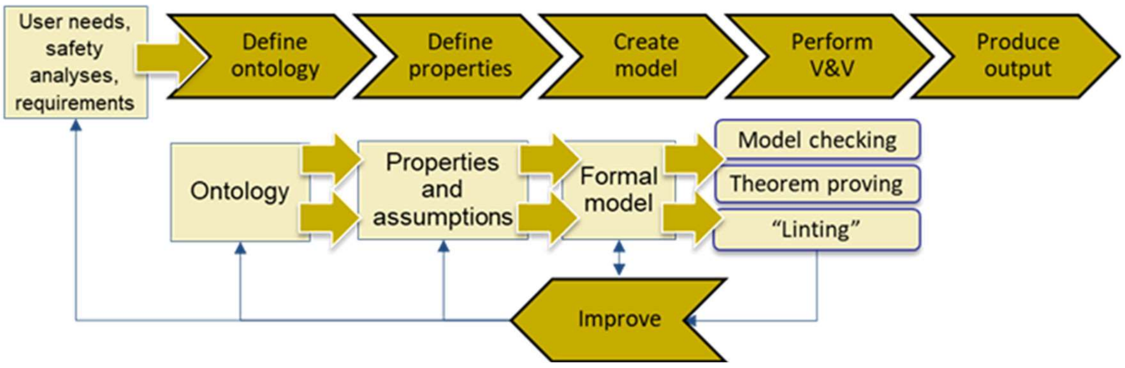

Figure 12-1: Generic process for FMs application

The following list briefly describes each step.

Define ontology. A conceptualisation of the domain structure which reflects the common view is defined, so that multiple stakeholders understand it in the same way. This includes relevant domain concepts, entities, ideas and their relations, which should be (a) well-chosen for the FM application purpose, as well as (b) sufficiently defined to be generally understood, and explicitly documented for later reference. Ontologies may range from a simple glossary to a formal ontology that allows automated reasoning.

Define properties. The relevant principles, requirements and assumptions for a system, its environment and its system parts are identified and defined in an informal language, using the concepts in the ontology. The aim is to formally prove the properties later and/or use them as part of such proofs. Various categories of properties exist, such as safety properties, environment assumptions, properties of configuration data, etc.

Create model. Some or all the defined properties are formalised in a chosen formal language to create a model. This may take the shape of a formal specification and/or a formal system model, and may be textual and/or graphical. The model can be created manually, or be automatically created (in whole or in part) by a tool from a specification, design, or implementation. Model creation should avoid unnecessary model complexity [11].

Perform V&V. The model is examined performing a range of verification and validation tasks such as manual review, simulation/testing, verification of mathematical consistency, generic and specific properties. Some of these tasks involve using formal techniques such as model checking, theorem proving or linting; some of them may require preparatory work such as the creation of configuration/test data or environment models.

Improve. Input material, ontology, properties, and model are adapted according to insights and learning, in particular from the two previous steps. This may include corrections, completions, but also improvements like simplifications, and may lead to iteration of previous steps.

Produce output. Depending on the FM application and the project context, artefacts for documentation (e.g. a V&V report), for later reuse (e.g. the model created) or as final result (e.g. improved requirements, implementation code generated from model) are produced or collected. Moreover, a successful FM application usually results in deeper knowledge for the (type of) system at hand, which may be incorporated in artefacts produced, distilled into guidelines or just remain as personal knowledge.

Finally, the guidebook points out and justifies that in general FMs work best when

- there is a clearly defined purpose (and system type) for the FM application and related best practices are followed,
- the scope includes the early stages of the system life cycle,
- staff possesses a complementary combination of skills, and
- learning from FMs application is reused "next time".

12.3.8 Example FMs Application

WP10 of the X2Rail-5 project creates a FMs case study demonstrator based on the standard principles and requirements for ETCS Level 3 Trackside with Moving Block (L3 Trackside) [12], recently issued by WP4. The purpose is to perform V&V in the first life cycle phase HL1 (see Section "FMs in the system life cycle"), by applying the generic FMs process (see Section "Applying FMs"). The case study will be more extensively described in WP10 deliverables at the end of X2Rail-5, and relate to the recommendations, and general/specific techniques in the FMs guidebook. It corresponds to the System Inception purpose (see Section "Example purposes for using FMs").

The case study considers L3 Trackside to be a self-contained subsystem with an event-based execution model. Compared to ETCS L2, a difference is that no interlocking subsystem is assumed (introducing an interlocking subsystem in the architecture "belongs to" the next high-level life cycle phase HL2). An event-based execution model was a natural choice to model the requirements in [12] for phase HL1. Events include ETCS messages (via radio), commands from traffic management, and status information from wayside objects (e.g., points, trackside train detection).

The ontology basis for the FMs demonstrator defines concepts in [12] as different types of objects, areas, and paths. Areas and paths are concepts that most people understand without difficulty, introduced to add more stringency (e.g., "Track Status Area" in [12] sometimes refers to an area, and sometimes to a path). A key concept of the ontology is the "dynamic path", a path whose extent is determined dynamically. Dynamic paths are "first-class citizen" objects in the overall property-based reasoning for moving block requirements. The dynamic path concept itself is also relevant for FMs application and enables abstractions that are used, such as disregarding balise groups and actual distances in train position reports.

Properties defined for the case study include different types of assumptions (for adjacent systems, scope limitations, simplifying assumptions) and requirements that implementations of L3 Trackside shall satisfy. Formal verification is used for two complementary V&V purposes: (1) prove critical system properties, and (2) validate behaviour that the system allows. The latter is done by formal verification of dedicated properties, expressing that a sequence of system states / events is not reachable (is impossible). If such a property is proved true, then no sequence as specified is possible; if the property is falsifiable, the generated counter example illustrates a possible sequence. This type of properties is used to validate the behaviour. The following specific techniques (see end of Section "Understanding what FMs are") are used:

- Automated creation of a formal model (in the "High Level Language" HLL [13])
- Formal verification, with counter example generation (using a model checker for HLL)
- Visualisation of ontology concepts in counter examples, in a graphical track layout
- Debug capabilities, to isolate and understand the cause of a dynamic situation involving the properties defined

From the point of view of using FMs, defining properties and performing V&V for the case study does not present any principal technical challenges. The main challenges relate to that [12] provides a partial view of L3 Trackside requirements, as it builds on L2, and non-harmonised operational procedures (e.g. Shunting, Staff Responsible mode), requiring domain expertise. For this reason, learning about ETCS (to a degree) was required. Even so, informal property-based reasoning by FMs practitioners (using earlier versions of [12]) raised relevant questions for a small number of question & answer sessions with authors. Already this informal reasoning helped improve the quality of requirements, matching previous experience on FMs benefits [10]; it leads to increased quality at requirement stage, which is a big positive effect, as requirements issues are cheaper to address at this stage, and because the reuse factor is the largest (cf. HL1 in Table 12-2).

To what extent FMs-based V&V in the demonstrator that WP10 works on can help improve quality of requirements further, remains to be evaluated (work in progress at the time of writing).

12.4 Summary, Conclusions and Outlook

Shift2Rail TD2.7 decided to structure the guidebook based on describing:

- The generic life cycle phases that are relevant for FMs application, and the opportunities and recommendations for FMs use in each of them.
- The importance of having a clearly defined purpose for using FMs in a project and of making informed decisions when planning FMs use for that purpose (e.g., staffing, picking FMs, processes, tools, …).
- The generic FMs application process (independent of life cycle phase, specific FMs languages and tools used), in terms of its steps (activities).
- Recommendations and guidance for FMs application, for some typical purposes. Recommendations relate to the life cycle phases and the generic FMs application process, focused on being "FMs-friendly".

Some conclusions from work on the guidebook are that

- production of the guidebook was not an easy task; many discussions on FMs topics were held, such as on the purpose of using FMs, the benefits attainable and different styles and processes in using FMs (in the past, and for foreseeable future). A common view had to be agreed while personal backgrounds varied (expectations on, attitude towards, knowledge of, experience with and preferences for FMs).
- even though FMs have been used extensively for railway signalling in the past, there is a lack of reference structure to categorise different projects applying FMs, enable comparison of them and to better understand them in terms of the generic FMs application process. This may be a gap that the guidebook aims to fill.
- there is a general need for case study descriptions of projects that have used FMs, in terms of the guidebook structure and vocabulary. Describing representative FM applications in the guidebook would have required more resources than available (and bear the risk of becoming outdated).
- most known industrial projects using FMs in the past have related to phases HL3 and HL4, or from HL1 through HL4. Even though many have been successful (you always find errors if you perform formal verification), more benefits due to FMs seem possible by focussing on HL1 – to achieve high-quality requirements – as well as HL2 – to achieve compliant system architectures.
- the need for FMs experts will remain despite the guidebook (just like in any engineering discipline). The guidebook hopefully enables fruitful communication with FMs experts, but cannot fully replace expert judgement in specific project situations e.g. regarding suitability of a particular FM or tool, analysis of the impact of project decisions, resolving of conflicting requirements on FMs application, and understanding FM results in depth.

The guidebook is currently being completed and will be made publicly available on X2Rail-5 project website [14]. It will contain more details than could be included in the current extended abstract. As the guidebook does not provide case study reports on FMs use, WP10 aims to describe its ongoing moving block case study in terms of the guidebook structure in upcoming project deliverables, beyond what has been presented in Section "Example FMs Application", thereby validating the guidebook concepts. Any feedback on the guidebook creation and contents to the authors of the extended abstract or to the Shift2Rail TD2.7 group would be welcome.

12.5 References

[1] A. Fantechi (2013): Twenty-Five Years of Formal Methods and Railways: What Next? In: Software Engineering and Formal Methods. LNCS, Vol. 8368, Springer, pp 167–183, https://doi.org/10.1007/978-3-319-05032-4_13

[2] A. Ferrari and M. H. ter Beek (2023): Formal Methods in Railways: a Systematic Mapping Study. ACM Computing Surveys, Vol. 55, Issue 4, Article no. 69, pp 1–37, https://doi.org/10.1145/3520480

[3] C. Limbrée, Q. Cappart, C. Pecheur and S. Tonetta (2016): Verification of Railway Interlocking – Compositional Approach with OCRA. In: T. Lecomte, R. Pinger, A. Romanovsky (eds): Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification. RSSRail 2016. LNCS, Vol. 9707, Springer, pp 134–149, https://doi.org/10.1007/978-3-319-33951-1_10

[4] P. Behm, P. Benoit, A. Faivre and J.-M. Meynadier (1999): Météor: A Successful Application of B in a Large Project. In J. M. Wing, J. Woodcock and J. Davies (Eds.): Proceedings of the Wold Congress on Formal Methods in the Development of Computing Systems (FM '99), Vol. I. Springer-Verlag, London, UK, 369–387

[5] EN 50128:2011 Railway applications – Communication, signalling and processing systems – Software for railway control and protection systems. CENELEC CLC/TC 9X standard, 2011-06

[6] X2Rail-2 WP5 (2018): Formal Methods (Taxonomy and Survey), Proposed Methods and Applications. Deliverable 5.1 of the X2Rail-2 project, available from https://projects.shift2rail.org/s2r_ip2_n.aspx?p=X2RAIL-2, accessed on 14/02/2023

[7] EULYNX initiative (2023): EULYNX website. URL: https://eulynx.eu/, accessed on 06/02/2023

[8] ERTMS Users Group (2023): Reference CCS Architecture subpage on User Group website. URL: https://ertms.be/workgroups/ccs_architecture, accessed on 06/02/2023

[9] EN 50126:2017 Railway Applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS) - Part 1: Generic RAMS Process. CENELEC CLC/TC 9X standard, 2017

[10] D. M. Berry (2002): Formal methods: the very idea, some thoughts about why they work when they work. Science of Computer Programming, Vol. 42, Issue 1, pp 11–27, https://doi.org/10.1016/S0167-6423(01)00026-0

[11] J. F. Groote, T. W. D. M. Kouters and A. Osaiweran (2015): Specification guidelines to avoid the state space explosion problem. Softw. Test. Verif. Reliab. 2015;25:4–33, https://doi.org/10.1002/stvr.1536

[12] X2Rail-5 WP4 (2022): Moving Block Specification. Deliverable D4.1 of the X2Rail-5 project, to be published on [14]

[13] J. Ordioni, N. Breton, J.-L. Colaço (2018). HLL v.2.7 Modelling Language Specification. STF-16-01805, RATP, https://hal.archives-ouvertes.fr/hal-01799749

[14] Shift2Rail Joint Undertaking (2023): X2Rail-5 project subpages on Shift2Rail website. URL: https://projects.shift2rail.org/s2r_ip2_n.aspx?p=X2RAIL-5, accessed on 07/02/2023

12.6 Acknowledgements

The X2-Rail-5 project has received funding from the Shift2Rail Joint Undertaking (JU) under grant agreement No 101014520. The JU receives support from the European Union's Horizon 2020 research and innovation programme and the Shift2Rail JU members other than the Union.

The paper reflects only the authors' view, the JU is not responsible for any use that may be made of the information it contains.

Europe's Rail

Horizon 2020
European Union Funding
for Research & Innovation

12.7 Authors

Daniel Schwencke graduated in Computer Science and received his PhD from the Technical University of Braunschweig. In 2011 he joined the DLR Institute of Transportation Systems as researcher, working in the field of railway safety, including system development and authorisation processes as well as human reliability. Since 2015 he is part of the verification and validation department, conducting research on model-based testing of signalling systems and test automation. He is involved in the European X2Rail-5 and R2DATO projects.

E-Mail: daniel.schwencke@dlr.de

Arne Borälv studied Computer Science at Uppsala University 1991-1995 and joined Prover Technology in 1995 and has worked on industrial application of formal verification since then, including in railways, automotive, avionics, and electronic design automation (EDA). He is involved in R&D within the European X2Rail-5 and R2DATO projects on behalf of Trafikverket (The Swedish Transport Administration).

E-Mail: arne.boralv@prover.com

Fernando Mejia graduated in Computer Science and received his PhD from the Paris University in 1982. He has worked on the industrial application of formal methods since then. First for electronic and computer manufacturers, and then, for 30 years for Alstom, a railway signalling manufacturer. He is involved in the European X2Rail-5 project as an independent consultant for Alstom.

E-Mail: fernando.mejia-ext@alstomgroup.com