Survey on Formal Methods and Tools in Railways
Technical Report on the activities performed within ASTRail, Deliverable D4.1

Technical Report ID	**ASTRail-D4.1**
Title	**Survey on Formal Methods and Tools in Railways**
Work Package	**WP4**
Dissemination Level	**PUBLIC**
Version	**4.1**
Date	**2018-05-31**
Status	**Final Version for Review**

Lead Author	**Alessio Ferrari (ISTI-CNR) Email: alessio.ferrari@isti.cnr.it**
Main Contributors	**ISTI-CNR: Maurice H. ter Beek, Franco Mazzanti, Davide Basile, Alessandro Fantechi, Stefania Gnesi** **SIRTI s.p.a: Andrea Piattino, Bruno Sturani and Daniele Trentini**

Published by the ASTRail Consortium

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 777561
Call identifier: H2020-S2RJU-2017 | Topic: S2R-OC-IP2-01-2017 – Operational conditions of the signalling and automation systems; signalling system hazard analysis and GNSS SIS characterization along with Formal Method application in railway field

Document History

Version	Date	Author(s)	Description
0.1	2018-03-28	CNR	TOC
1.0	2018-04-13	CNR	First Version, with Chapter 1, 3 and 5
2.0	2018-04-19	CNR, SIRTI	Second Version, with modifications from SIRTI
3.0	2018-05-16	CNR	Final Version, with Chapter 2, 4, 6, 7, 8
4.0	2018-05-27	CNR	Final Version, which incorporates the review from ISMB
4.1	2018-05-31	ISMB	Legal notice added. Minor formatting

Legal Notice

Table of Contents

1 Introduction

1.1 Background

The term formal method identifies a set of mathematically rigorous practices to support the development of software intensive systems [CLA96] [BOC09]. It is common to consider three application levels at which formal methods can be employed for the development of a system: formal specification, formal development and machine checked proof. Formal methods that embrace all the three levels are development frameworks composed by a formal language, a verification strategy and a set of tools to translate a specification into an implementation in a rigorous manner [CRA95]. In practice real-world formal methods address only one or two of the application levels [MON12], for example with a formal specification language that can be used to provide a model for the system, and a tool-supported verification strategy that allows the user to prove properties on the model.

Formal methods have been largely experimented in industry for the development of safety-critical and mission critical products [WOD12]. Notable industrial cases on the usage of formal methods are the Maeslant Kering storm surge barrier control system [TWC01], where both the Z and the Promela formal notations have been used, and the Paris Metro on-board equipment [BBFM99], where the B-method has been employed. Transportation in general and railways in particular are domains in which formal methods have been largely experimented and applied [WOD12]. Research and industrial experiences concerning formal methods applications to railway systems' development have been published for more than thirty years [FAN13] [BGK18], and scientific publications in this field are increasing, showing that the interest in formal methods is still raising, but also indicating that more research is needed for a full industrial uptake of formal methods in railways.

1.2 Purpose and Scope

Despite the quite long story of successful application of formal methods in the railway domain, it cannot yet be said that a single mature technology has emerged. Indeed, any proposed method or technique that goes under the umbrella of formal methods varies in its suitability and applicability to different stages of the signalling system development, and to different subdomains of railway signalling (interlocking, ATP, ATS, *etc.*).

This **Work Package 4** of the ASTRail project aims to identify, on the basis of an analysis of the state of the art, of the past experiences of the involved partners and on work done in previous projects, the candidate set of formal and semi-formal techniques that appear as the most adequate to be used in the different phases of the conception, design and development of a railway equipment in general, and of the class of signalling equipment that is the subject of this project in particular. In the following, when we will use the general term formal method, we will implicitly include also semi-formal methods, i.e., those methods that use languages for which the semantics is not formally defined but depends on its execution engine. Furthermore, given that, in practice, a formal method always needs a support tool to be practically applicable, we will use the terms formal methods and formal tools interchangeably.

1.3 Overview

Figure 1 presents the overall approach applied in the context of this deliverable. To address the goal of identifying the most mature formal / semi-formal languages and tools to be applied for the development of railway systems, we first performed a benchmarking task (Task T4.1), by gathering information from three different sources: **Scientific Literature**, information from other **Projects**, and **Railway Practitioners**.

Information from these sources was gathered through a **Systematic Literature Review** (SLR), a **Projects Review** and a **Survey** submitted to practitioners in the form of a questionnaire.
The information was used to identify a set of main formal and semi-formal tools that appear to have been used in the railway domain (**Relevant Tools**). Specifically, scientific literature was used as a primary source, since it provides more extensive information for guidance in the selection of relevant formal methods, while other projects and railway practitioners were used as sources to complement the information from the literature review. Furthermore, **Evaluation Criteria** for the different tools were defined based on collaboration between academic and industrial partners. These were applied to carefully evaluate the selected tools in a **Tool Review**.

After the benchmarking task, a ranking task was performed (Task T4.2). During this task, the SLR was used to produce a **Tool Selection Support Matrix (Annex 1)**, which supports in the identification of the possible tools to be used depending on the specific railway system to be developed, and depending on the life-cycle phase to address. Furthermore, a **Ranking Matrix (Annex 2)** was defined for the different tools based on the tool review, and a **Tool Evaluation** document **(Annex 3)**, which provides details about the evaluated tools. The Ranking Matrix aims to support the selection of a formal or semi-formal tool for the railway problem at hand, based on specific preferences selected by the user of the matrix, concerning different evaluation criteria (e.g., functionalities supported by the tool, flexibility, usability) and guided by the information from the Tool Selection Support Matrix. The Tool Evaluation document provides more details to perform a more informed selection.

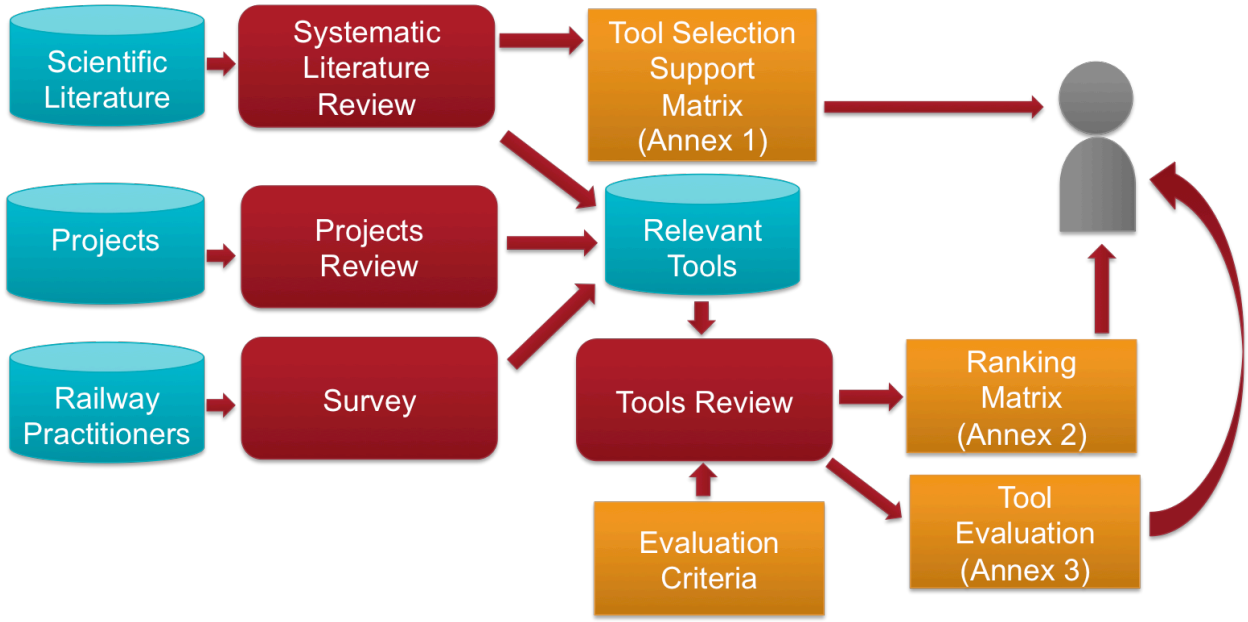


Figure 1 Overview

The remainder of this deliverable is organised as follows:

- Section 2 presents some background on formal and semi-formal methods, defining the concepts that are useful to understand the rest of the deliverable;
- Section 3 reports the results of the SLR conducted to identify the most used formal and semi-formal method in railways;

- Section 4 reports the results of the projects review, conducted to complement the results from the SLR;
- Section 5 reports the results of the survey with practitioners, aimed at integrating the information from the SLR and at identifying relevant evaluation criteria to be used for the tool review;
- Section 6 reports the activity of tools review, conducted to provide a fine-grained evaluation of the different relevant tools identified through the SLR, the projects review and the survey;
- Section 7 reports the results of the ranking task, presenting the usage of the Tool Selection Support Matrix and the Ranking Matrix to be used for the selection of formal methods;
- Section 8 provides conclusion and final remarks.

1.4 Related documents

ID	Title	Reference	Version	Date
[RD.1]	Annex 1 - Tool Selection Support Matrix (see Excel source: https://goo.gl/TqGQx5)	Annex 1	1.0	2018/16/05
[RD.2]	Annex 2 – Ranking Matrix (see Excel source: https://goo.gl/1hB5qz)	Annex 2	1.0	2018/16/05
[RD.3]	Annex 3 – Tool Evaluation	Annex 3	1.0	2018/16/05

2 Background on Formal and Semi-formal Methods

2.1 What are Formal and Semi-formal Methods and Tools

The central problem of formal methods is to be able to guarantee the behaviour of a given computing system following some rigorous approach. At the heart of formal methods one finds the notion of *specification*. A specification is a model of a system that contains a description of its desired behaviour—*what* is to be implemented, as opposed to *how*. This specification may be totally abstract, in which case the model *is* the description of the behaviour, or it may be more operational, in which case the description is somehow contained, or implied, by the model. In this context, the central problem can be seen as being divided in the following two aspects:

1. How to enforce, at the specification level, the desired behaviour? This is called the *model validation* problem.
2. How to obtain, from a specification, an implementation with the same behaviour? Or alternatively, given an implementation, how can it be guaranteed that it has the same behaviour as the specification? This is the *formal relation between specifications and implementations* problem.

The different families of formal methods cover a wide range of approaches to answer these two questions. This variety encompasses the study of specifications by *animation* or *simulation*, by *transformation*, or by *proving properties*. Analogously, implementations may be *derived* from specifications, or else they may be *guaranteed to be correct* with respect to them, either by *construction*, or by *verification*, i.e., by presenting a *formal proof*.

We introduce below some concepts and notions that characterise the application of formal methods and tools in the design of software systems [AFPM11] [OR17].

Model-based development puts a conceptual system model at the centre of the development process, from requirements engineering, through (model-based) design, to *model-based testing* -- which is characterised by test cases derived from models rather than from source code -- and possibly *code generation* -- which automatically translates system models to source code. Consistency among the models used in the various phases is ensured through model transformation and refinement. *Model transformation* can be seen as the automatic generation of a target model from a source model based on a transformation definition.

Refinement concerns the verifiable step-wise transformation of an abstract (high-level) formal specification into a concrete (low-level) executable program, such that each step increases the level of detail (e.g., which algorithm or which data type to implement).

Synthesis aims to automatically construct a system or program that is guaranteed to satisfy a given (high-level) specification.

Type checking offers a means to analyse the well-formedness of a model (or source code) with respect to its meta-model, which is formally specified as a type system that all models must conform to. This is a form of *static analysis,* i.e. check to be performed without executing the program/model.

Model checking is a technique to automatically and exhaustively verify whether a formal model of a system satisfies its specification, expressed as properties in a (temporal) logic. With respect to testing, model checking thus exhaustively verifies all possible behaviour, typically providing a counterexample in case a property is not satisfied. *Reachability analysis* is limited to verifying whether the behaviour allows to reach a certain system state.

Statistical Model Checking this technique is similar to model checking in that it uses a formal specification of a system in a specific (probabilistic) formalism and the definition of properties of interest using a particular

temporal logic formalism. However, Statistical Model Checking does not exhaustively generates the whole state-space of a system, but it rather evaluates the properties to be verified by systematically simulating the system until a specific degree of confidence is reached.

Theorem proving is a deductive approach to prove the correctness of logical formulas by applying inference rules to them, either interactively or automatically, resulting in a proof script listing the deductive reasoning (for inspection by humans). Model checking and theorem proving generally do not scale to huge systems. In such cases, (interactive) *simulation* (i.e., a sample path or execution) of the system model's behaviour can still provide valuable insights.

Two well-known techniques for automated theorem proving are SAT solving and SMT solving. *SAT solving* aims to provide a variable assignment such that a given propositional logic formula is satisfied or else provide a subset of clauses that cannot be satisfied. *SMT solving* does the same for the richer first-order logic, which allows more complex expressions than propositional variables (involving constants, functions and predicates). While SAT solving and SMT solving are NP-hard problems, there are numerous instances of SAT/SMT problems over thousands of variables that are easily solved by current solvers. In fact, modern SAT/SMT solvers are very fast most of the time.

Abstract interpretation is a formal static (program) analysis used to debug a version of the source code by analysing the effect of statements on an abstract system model that over-approximates the actual system's behaviour. This makes the analysis simpler, at the cost of incompleteness (i.e., not every system property is also satisfied by the abstract system) while preserving soundness (i.e., every property satisfied by the abstract system corresponds to an actual system property).

Prototyping is the activity of producing prototypes of software applications or tools, i.e., not yet complete versions of the software program under development, as is common for other engineering disciplines.

Contract-based design is a correctness-by-construction paradigm according to which system components at different levels of abstraction are characterised by contracts in the form of the properties that these components have to guarantee and the assumptions that have to be satisfied by their environment.

3 Systematic Literature Review

3.1 Systematic Literature Review (SLR) Methodology

The Systematic Literature Review (SLR) approach is an empirical technique for retrieving information from scientific literature [BP06][KBB09]. This empirically grounded technique is recommended for retrieving scientific papers from online databases that speak about a certain topic of interest for the review, and it is widely recognised and used in the software engineering field.

In particular, an SLR consists of [KBB09]:

1. Definition of a set of research questions that the SLR aims to answer;
2. Definition of a search string to be used to retrieve relevant publication from scientific databases, such as the ACM Digital Library or SpringerLink;
3. Definition of a set of inclusion and exclusion criteria, aimed at selecting the relevant papers for the purpose of the SLR;
4. Definition of a study selection procedure, to apply the criteria;
5. Definition of a quality checklist, to assess papers that are relevant and have high scientific quality;

6. Data extraction strategy, to extract information relevant to the research questions from the selected publications.

In the following, we describe how the SLR technique has been applied in the context of the ASTRail project to identify papers concerning the application of formal or semi-formal methods in the development of railway products.

3.2 Application of the SLR Methodology

3.2.1 Goal and Research Questions

The main goal of our SLR is as follows:

Goal: Identify which formal and semi-formal methods are the most mature for application in the railway industry

To address this goal, we first aim to provide a characterisation of primary studies concerning applications of formal and semi-formal methods in the railway domain. Then, we plan to identify which methods are used, in which phase of the development process, and for what types of systems. Finally, we aim to identify the degree of maturity of the methods for industrial application. Therefore, from our goal, we derive the following research questions (RQs):

- RQ1: What primary studies have been conducted about applications of formal and semi-formal methods in the railway domain?
  - RQ1.1: Which types of methods are used?
  - RQ1.2: In which phases of the system development are the methods applied?
  - RQ1.3: To which type of railway systems are the methods applied?
- RQ2: Which methods are the most mature for industrial application?

3.2.2 Search Strategy

The search strategy indicates the approach to be followed for retrieving papers. According to Kitchenham et al. [KBB09] it is normally recommended to perform a primary search strategy, based on a search string and the automatic retrieval of papers from scientific databases, followed by a secondary search strategy, based on manual search of relevant publications. In the context of this deliverable we performed solely the primary search strategy, due to resource constraints. However, the literature cited by the identified secondary studies -- other, non-systematic surveys -- has been cross-checked to identify possibly missing papers and assess that the study cited by these surveys are covered by the studies identified with the primary search strategy. All the papers retrieved are evaluated regardless of their year of publication.

Our **primary search strategy** consists of the following steps:

1. Define a set of major terms to be searched
2. Perform a pilot test using the major terms, to identify synonyms, alternative spellings, or similar terms to be searched
3. Connect the resulting terms using boolean operators
4. Select a range of online databases to be used
5. Import and manage the retrieved studies with Zotero[1]

[1] https://www.zotero.org

Our **search terms** stem from the research questions, and can be categorised according to two dimensions, with associated keywords:

1. Object: "formal" OR "model check*" OR "model based" OR "model driven" OR "theorem prov*" OR "static analysis"
2. Domain: "railway*" OR "CBTC" OR "ERTMS" OR "ETCS" OR "interlocking" OR "automatic train" OR "train control" OR "metro" OR "CENELEC"

The keywords were selected based on brainstorming among the participants, and pilot test (see below). The set of OR-keywords in the **Object** dimension and the set of OR-keywords in the **Domain** dimension are combined with the AND operator to form the final search string. Moreover, keywords including the symbol *, for example "railway*", stands for all possible keywords with prefix "railway".

Online databases to be searched are the typical databases for scientific computer science and engineering literature [KBB09], and are:

- ACM Digital Library[2]
- IEEE Explore Digital Library[3]
- SpringerLink[4]
- ScienceDirect[5]

Pilot Test A pilot test for the SLR was performed using a first version of the search string on four online databases. We applied inclusion/exclusion criteria on a sample of first 10 papers from the results of all four databases. Two researchers performed the selection process in parallel, over the top-10 results of each database. The Pilot Test enabled to assess that, among the 40 studies considered, 24 of them resulted to be relevant for the RQs. This leads to a precision of 60%, which we consider acceptable for our goal. Recall cannot be evaluated in the Pilot Test. Besides being useful for assessing the effectiveness and refining the pilot string, the pilot test was used to refine the SLR procedure adopted and described in the following sections.

3.2.3 Study Selection Criteria and Procedures

Study selection criteria are used to determine which studies are included in, or excluded from, a systematic review. These criteria have been piloted on a subset of primary studies.

Inclusion Criteria

- The study presents an application of formal or semi-formal method, including model-based development methods, to the development of railway systems
- The study is mainly concerned with the development of railway systems for signalling and control
- The study comes from an acceptable source such as a peer-reviewed scientific journal, conference, symposium, or workshop
- The study is written in English language

Exclusion Criteria

- The study does not use a formal or semi-formal method

[2] https://dl.acm.org
[3] https://ieeexplore.ieee.org/Xplore/home.jsp
[4] https://link.springer.com
[5] https://www.sciencedirect.com

- The study does not apply a method to the railway domain
- The study uses a railway problem as a benchmark to validate a tool
- The study is concerned with the quantitative assessment of reliability, availability or maintainability requirements expressed in quantitative form
- The type of study is a secondary study (i.e., a survey)*
- The type of study is a book
- The study did not undergo a peer-review process
- The study has been published in another, extended form
- The study has the form of editorial, abstract, keynote, poster or a short paper (body less than 4 pages)

* NOTE: Secondary studies will be retained for cross-referencing.

The selection procedure for the study is as follows:

1. The researcher inspects the title, abstract and venue of the study, and will apply the selection criteria. In case it is not possible to establish inclusion or exclusion based on the aforementioned information, the introduction and conclusion sections will be inspected;
2. The studies that fulfil the selection criteria will be selected as relevant studies and will be assigned a unique number;
3. The studies that appear as duplicated in different sources will be considered as a single study, with the same unique number, and the different sources will be retained.

Two researchers apply the study selection procedure outlined on separate subsets of the retrieved studies. When a study includes the researcher himself among the authors, the other researcher performs the selection.

3.2.4 Study Quality Assessment

The primary assessment strategy is based on the following checklist, which is adapted to our context from [DD08] and employed also by [CB11]:

1. Is the paper based on research (or is it merely a ‘‘lessons learned’’ report based on expert opinion)?
2. Is there a clear statement of the aims of the research?
3. Is there an adequate description of the context in which the research was carried out?
4. Was the research design appropriate to address the aims of the research?
5. Is it clear the type of railway system considered?
6. Is it clear the set of formal and semi-formal tools used?
7. Is it clear the phase of the process in which the approach is applied?
8. Is it clear the involvement of industrial partners?
9. Is the railway case sufficiently realistic and representative?
10. Is there a clear statement of findings?
11. Are threats to validity considered?
12. Is the study of value for research?
13. Is the study of value for practice?

We use a ternary (‘‘Yes’’= 1, ‘‘Partial’’=0.5 or ‘‘No’’=0) scale to grade the reviewed studies on each question reported in the checklist. If a study does not reach a grade of 6, it is not considered for the Data Extraction phase, due to poor quality. The information acquired from the quality assessment performed by means of the quality checklist is used also in the Data Extraction phase, in order to establish the degree of quality of the evaluated studies.

3.2.5 Data Extraction

Data extraction is conducted by two researchers, referred in the following as extractors. Relevant publications are partitioned into two sets, and each extractor extracts data from one set. Each extractor reviews the extracted data for the other set. In case of disagreement, another researcher, referred as supervisor, will guide towards a decision on the data to be recorded. When one of the extractor is author of a relevant study, the other extractor conducts the data extraction. The data is recorded in the form of spreadsheets to ease the analysis.

Data to be extracted are as follows:

PUBLICATION DETAILS

- Title
- Authors
- Publication year
- Full citation

Publication details are not reported in the Tool Selection Support Matrix in Annex 1. Instead, a Web link to the original publication is reported.

CONTEXT DESCRIPTION

Degree of formality of the method(s):

- F: Formal, when all the methods used by the paper are formal
- SF: Semi-formal, when all the methods used by the paper are semi-formal
- SFF: Semi-formal and Formal, when the paper uses a combination of formal and semi-formal methods

Name(s) of the technique(s) applied: the list of techniques applied in the paper (e.g., model-based development, model checking, theorem proving, etc.)

Name of the language(s): the name of the languages used for modelling in the context of the paper (e.g., UML, State Machines, etc.)

Name(s) of the support tool(s) used: the names of the tools used in the paper (e.g., SCADE, Simulink, UPPAAL)

Phase(s) of the system development addressed (from the CENELEC norms [CEN11]**):**

- Planning (P)
- Requirements (R)
- Arch.& Design (A)
- Component Design (D)
- Implementation (I)
- Testing (T)
- Integration (N)
- Validation (V)
- Maintenance (M)

Task(s):

- **SM: Specification / Modelling.** The paper addresses this task if the system is modelled with a textual or graphical, formal or semi-formal language.
- **SIM: Simulation.** The paper addresses this task if it presents the results achieved with graphical or textual simulation of a model.
- **ANA: Analysis of Specifications / Models.** The paper addresses this task if it presents the result of some formal analysis different from formal verification for the inspection or debugging of the model.
- **FV: Formal Verification.** The paper addresses this task if it performs any form of formal verification on the model.
- **REF: Refinement.** The paper addresses this task if it presents the application of a refinement strategy from a high-level model towards a lower level model.
- **MBT: Model-Based Testing (Test Generation / Oracle Specification).** The paper addresses this task if model-based testing is applied either to generate tests for the model or for the actual implementation, or as a specification of an oracle against which test results on the code are compared.
- **CG: Code Generation.** The paper addresses this task if source code is produced from the model.
- **SA: Static Analysis.** The paper addressed this task if a form of static analysis is applied on the source code.

Category of railway system:

- SA (Stand-alone system): if the system treated in the paper is a part of an interlocking system that can be treated independently, as for the case of Platform Screen Door Controllers, Railway Crossing Controllers, Axle Counters
- ERTMS/ETCS: if the system is part of an ERTMS/ETCS system
- CBTC: if the system is part of a CBTC system
- CTCS: if the system is part of a CTCS system
- NS (Non-standard Train Control and Management): if the system is part of a train control and management system that does not follow an international standard

Subcategory of Railway System:

- For Stand-alone systems (SA):
  - Name of the system (e.g., Platform Screen Door, Railway Crossing, Axle Counter, etc.)
- For ERTMS/ETCS, CBTC, CTCS, NS:
  - HLCL (High-level Control Logic and Communication): if the paper treats the high-level logic of the system, or the communication between two or more components
  - Name of the system (e.g., wayside ATP, on-board ATP, ATO, Radio Block Centre, etc.): if the paper treats solely one subsystem

EVALUATION INFORMATION

Type of Study and Evaluation (from [CB11]):

- Rigorous analysis (RA): Rigorous derivation and proof, suited for formal model
- **Case study (CS):** An empirical inquiry that investigates a contemporary phenomenon within its real-life context; when the boundaries between phenomenon and context are not clearly evident; and in which multiple sources of evidence are used

- **Discussion (DC):** Provided some qualitative, textual, opinion
- **Example (EX):** Authors describing an application and provide an example to assist in the description, but the example is “used to validate” or “evaluate” as far as the authors suggest
- **Experience Report (ER):** The result has been used on real examples, but not in the form of case studies or controlled experiments, the evidence of its use is collected informally or formally
- Field study (FS): Controlled experiment performed in industry settings
- Laboratory experiment with human subjects (LH): Identification of precise relationships between variables in a designed controlled environment using human subjects and quantitative techniques
- Laboratory experiment with software subjects (LS): A laboratory experiment to compare the performance of newly proposed system with other existing systems
- Simulation (SI): Execution of a system with artificial data, using a model of the real word

Among the previous works, the ones that have been identified in this SLR are the ones marked in **bold**.

Industrial Evaluation (adapted from [CB11])**:**

- NO: Not evaluated in industrial settings
- LAB: Industrial problem treated in laboratory settings
- IND: Industrial problem validated with railway experts
- DEV: Development of an industrial product

Authorship:

- A: only academic authors
- I: only industrial authors
- AI: academic and industrial authors

3.3 Results from the SLR

In this section, we summarise the most relevant results from the SLR. The search was conducted on the 7th of December, 2017, while the analysis and data extraction were performed in the following months. Figure 2 summarises the approach and the results. From the initial search, and a first analysis of the abstracts of the papers, we identified a set of 411 potentially relevant papers to use for data extraction. Given the large amount of literature, and given that the focus of ASTRail is not on interlocking products, we decided to focus solely on studies that do not deal exclusively with interlocking products (hence, studies that focus mainly on interlocking, 124, are excluded from our analysis). We manually analysed 294 papers to check their quality and identify any shorter versions of other papers in the set. We excluded 180 studies that appeared to have low quality, according to our quality checklist, or that appeared to be shorter versions of other papers that were present in the set. A set of 114 papers was therefore used for data extraction. Therefore, in the following, we report the extracted data from high-quality, and non-interlocking products. All the papers considered for data extraction, together with the extracted information, are available in Annex 1.

When appropriate, the statistics in the following sections will distinguish between the total number of papers considered in the review, and the papers that had either an industrial evaluation (identified with IND, in “Industrial Evaluation”, see Sect. 3.2.5) or that led to actually developed products (identified with DEV). These papers are considered as more important, since they show evidence of industrial maturity of a certain method or tool, and they provide support to answer RQ2.

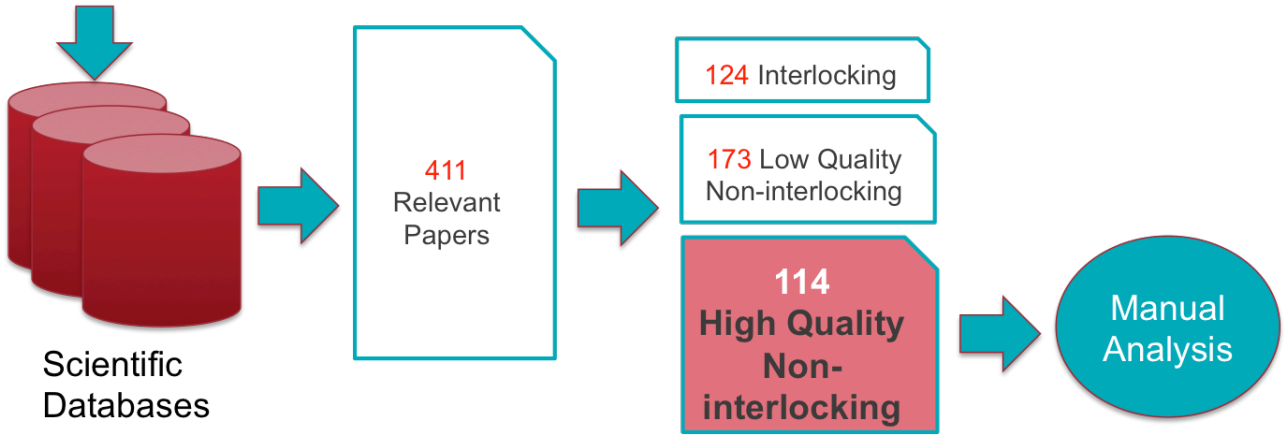


Figure 2 Summary of the approach followed for the SLR

3.3.1 Type of Study

Figure 3 summarises the different types of studies. The large majority of the works are Experience Reports (48%), i.e., descriptions of experiences in the application of formal methods to a railway problem. A relevant amount of works (31%) considers a railway case only as a motivating example, aimed at illustrating the formal technique discussed in the paper. Although presenting examples, these studies are however relevant to this survey, in that they may include examples that are extracted from works that describe industrial experiences in the usage of formal methods. Other works (15%, Example / Experience Report) are somehow extended examples, which make them closer to experience reports. A limited part of the studies are Discussion (3%), which includes description of lessons learned from different experiences with the usage of formal methods.

It should be noticed that very few studies (2%) can be considered Case Studies, i.e., rigorous empirical evaluations of experiences, and some studies (1%) are Experience Report with some partial rigorous evaluation in the form of a Case Study (1%, Experience Report / Case Study) although there is a large amount of research on formal methods in railways, the conduct and reporting of the research is still not scientifically mature, and, hence, empirically sound evidence is actually quite limited.

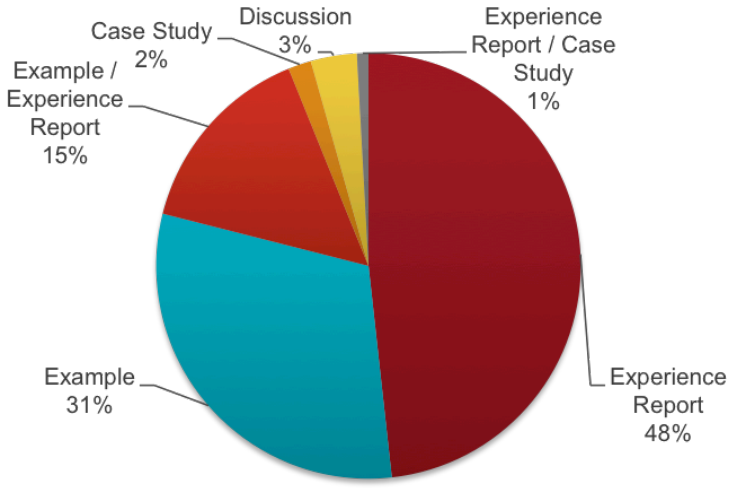


Figure 3 Type of study

3.3.2 Industrial Evaluation and Authorship

Figure 4 reports the results for the industrial involvement in the evaluation. The majority of the works (LAB, 59%) present an industrial problem, and validate the problem in a laboratory setting. In some cases (IND, 18%), the researchers applying formal methods to the railway problem validate the results with industrial partners. In other cases (DEV, 10%), the papers concern the development of a real product, and in few cases there is no industrial involvement (7%). Some papers are in-between the different classes (IND-DEV, 3%; LAB-IND, 3%). These results indicate that there is an attention to industrial problems, but the evidence on the application of formal methods within industry is still quite limited, or at least, few works have been published concerning the development of real railway products with the aid of formal methods.

The statistics in the following sections will distinguish between the total number of papers considered in the review, and the papers that are classified as IND, DEV and IND-DEV. The aggregate statistics about the papers in these three classes are presented with the format IND/DEV, and, in the text, the papers in the set will be referred as *industrial papers*.

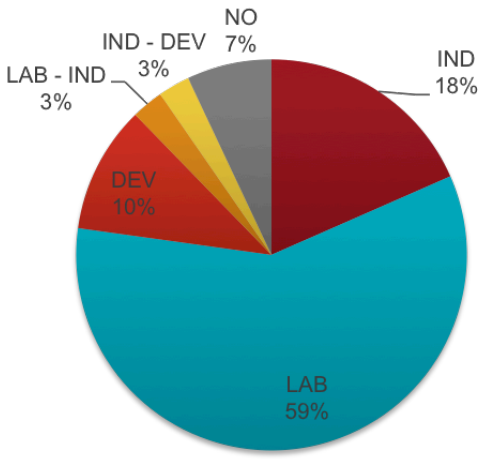


Figure 4 Industrial Evaluation

Figure 5 reports the results concerning the authorship of the papers. We see that most of the papers come from Academia (59%), part of them include authors from Academia and Industry (31%), and a non negligible set of papers has only author from Industry (10%). These results confirm the interest in formal methods by industrial practitioners, which, overall, appear in 41% or the papers, although, as one can expect from scientific publications, the large majority of works come from academic authors. We do not use the industrial authorship to separate industrial and academic papers in the following statistics, since we arguably believe that the level of evaluation conducted, presented in Figure 4, is a stronger indicator of the industrial representativeness of a certain work.

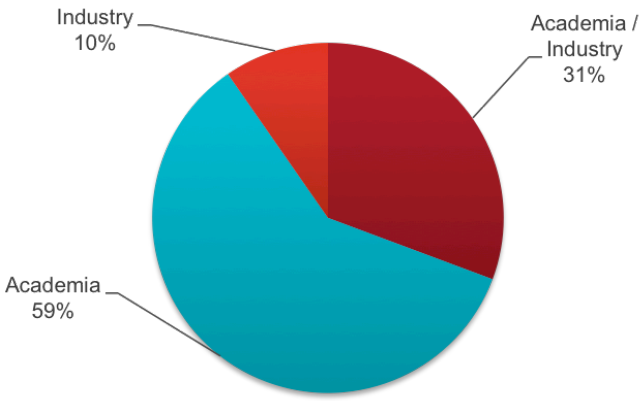


Figure 5 Authorship

3.3.3 Degree of Formality

Figure 6 presents the results for the degree of formality for the applied methodologies presented in the different papers. We see that the majority of the papers are Formal, i.e., using solely formal techniques; instead, a limited amount of them is exclusively Semi-formal, i.e., using solely semi-formal techniques or languages such as UML or SysML. Part of the works combine Semi-formal methods with Formal ones. It is interesting to notice that, considering industrial papers, these tend to use this combination more frequently than the other papers, suggesting that, for industry, it is important to use semi-formal techniques along with formal ones (e.g., for reasoning about a system and provide an high-level design in a semi-formal way, and then verify properties of the system by using formal verification techniques).

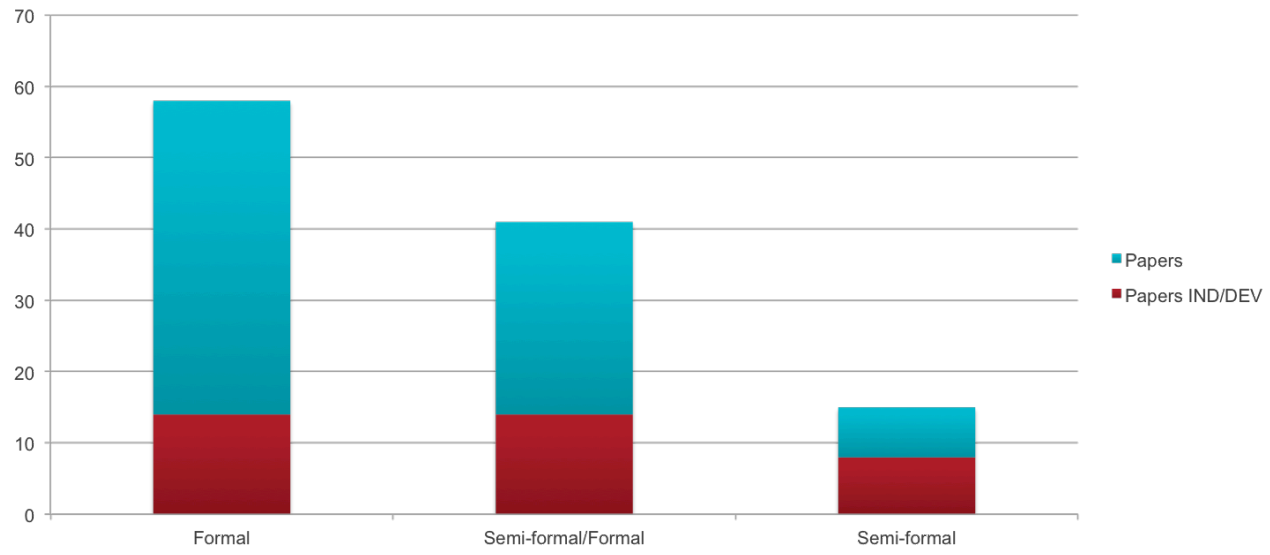


Figure 6 Formality

3.3.4 Techniques Applied

Figure 7 reports the results in terms of number of papers that apply a certain technique. It should be noticed that the inspected papers normally combine different techniques, hence, the different techniques are not mutually exclusive. For example, it is typical to combine model-based development with model checking, and model transformation can be considered as a subcategory of model-based development. However, with some loss of formality, we do not distinguish between categories and subcategories to give a simpler view of the results.

The most used techniques identified can be categorised as model-based development, i.e., approaches that use a formal or semi-formal model to support the development. These techniques are followed, in terms of frequency, by model checking techniques, which we recall to consist in systematically exploring the state space of a formal model of a system to verify that the model evolution respects certain temporal properties, as, e.g., no collision among trains, or no derailment. The third most used technique is theorem proving, which is a subfield of automated reasoning and mathematical logic dealing with proving mathematical theorems by computer programs. Highly frequent are also techniques for model transformation, since quite often the papers inspected consider a semi-formal model, e.g., UML, and translate this model into a formal model to be formally verified, e.g., with model checking. Model-based testing is also a frequently used technique, consisting in defining a high-level model of the system, and using this model for generating tests to be applied either on the model itself, or on the source code implementing the model. Refinement, consisting in applying a rigorous refinement process to a certain abstract design, is also relatively common. Other techniques, e.g., SAT, reachability analysis, etc., are also used, but are less frequent with respect to the other ones.

Industrial papers are mainly in line with academic ones in terms of frequency. An interesting little difference can be observed on techniques such as model-based testing and model transformation. The former is more frequent in industrial papers, while the latter is less frequent. This suggests that higher relevance is given in industry to the testing activities, while approaches for model transformation are more research works that still needs more development before they can be applied in industry.

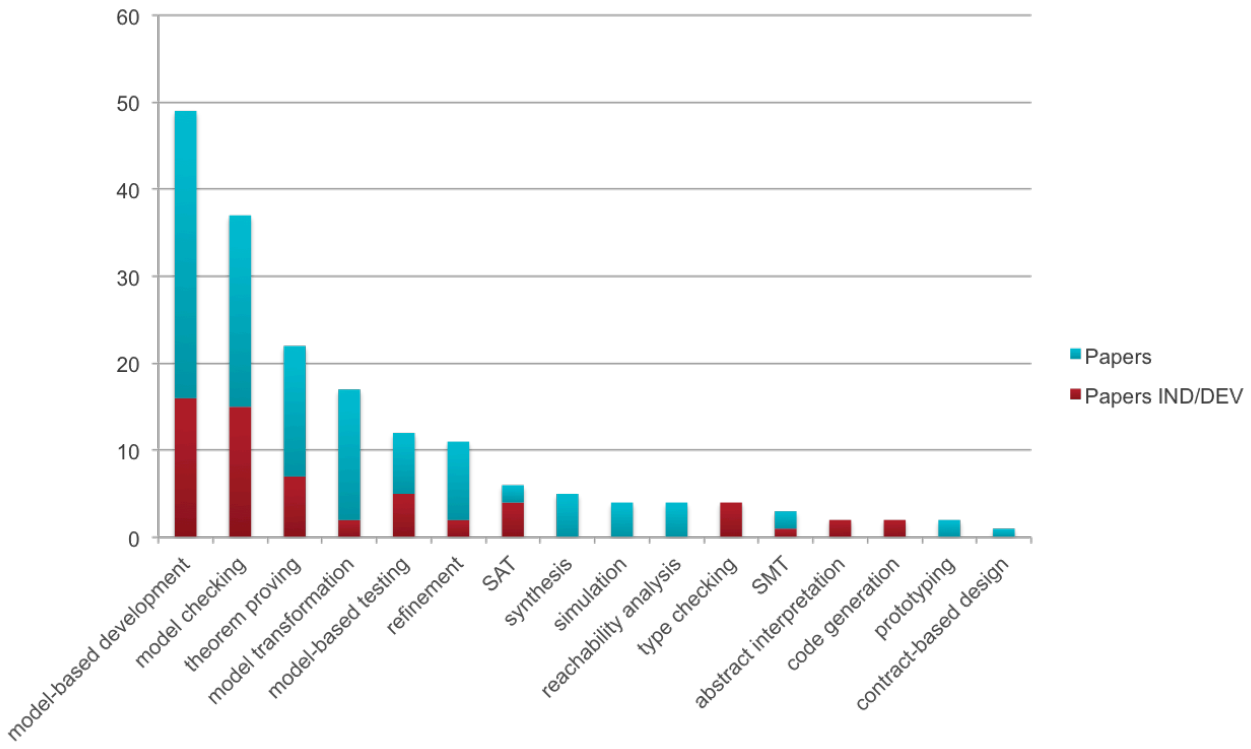


Figure 7 Techniques

3.3.5 Languages

Figure 8 reports the results in terms of number of papers that use certain semi-formal and formal languages. The list of languages is extensive, and, in the statistics, we do not report languages that appear only in one non-industrial paper. The most used input language, according to the papers analysed, is UML. This is a semi-formal language, which is often used in the early phases of system design, and it is normally translated into a formal language, as, e.g., the B language, in the considered studies. State Machines or Statecharts, in their different dialects, as for example Simulink/Stateflow, are also frequently used. Also more formal languages, such as Timed Automata, Petri Nets, CSP and Promela occur in a non-negligible amount of papers. However, these formal languages are mainly used in academic papers, and few industrial papers make use of them. Indeed, industrial papers tend to privilege State Machines, UML and B /Event B, or the SCADE language. It is also worth noticing that some industrial papers use several specific modelling languages, e.g., VDM, DSTM4Rail, etc., that are used only in the context of the paper, and are not used in purely academic papers.

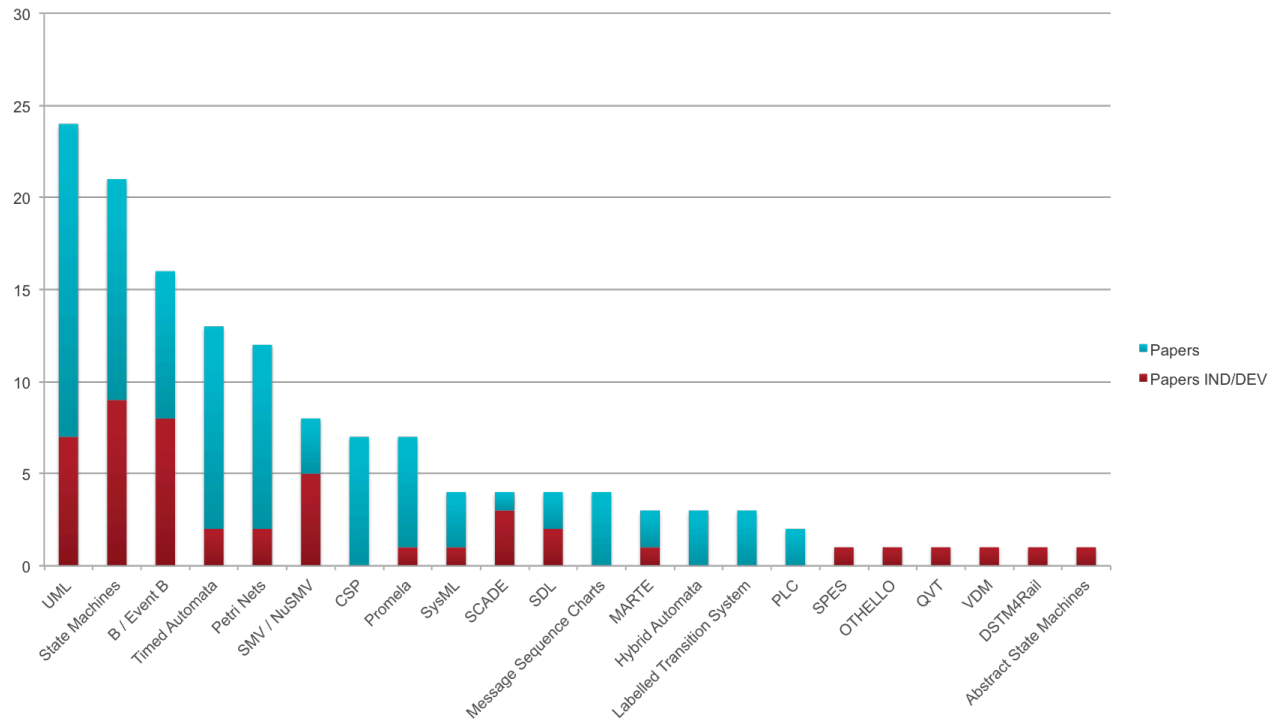


Figure 8 Languages

3.3.6 Support Tools

Figure 9 reports the results in terms of number of papers that use a certain support tool. The list of tools mentioned in these papers is extremely extensive and each paper uses a different combination of methods and tools. Therefore, in the statistics we consider solely those tools for which there are at least two papers using the tool.

The most used tools are those that belong to the B family: by summing up the contributions of Atelier B and ProB, we have 13 papers using these tools (Rodin is normally used in combination with Atelier B or ProB). By summing-up the contribution of the two tools, they also dominate in industrial works. These B-method tools are followed by Simulink, UPPAAL, NuSMV, SPIN and other tools. We do not report the complete list of identified tools, since this is particularly long, and we are here interested in identifying the most used tools for industrial works in railways.

Interestingly, tools such as UPPAAL and SPIN, which appear frequently in the papers, are less frequent in industrial works, in which, besides Atelier-B, we see a greater usage of NuSMV, Simulink, Statemate and SCADE. We also see that, among the industrial papers, NuSMV appears to be more frequently used than other tools, such as, e.g., Simulink, which is inherently more industry oriented. We argue that this may be related to the particular capability of NuSMV of deal with the formal verification of large, realistic systems. Simulink is more oriented to modelling and simulation, and its formal verification tool, Simulink Design Verifier, although used in industrial works, has been rarely used for formal verification of large systems, but more to sub-components. It should be however noticed that, in the inspected industrial papers, modelling and formal verification with NuSMV was not performed by railway practitioners, but by formal methods experts. This suggests that the usage of state-of-the-art formal verification in industry still requires the support of formal methods experts to be actually effective in practice.

Overall, we notice that there is a large fragmentation in the papers in terms of used tools, and even the most used tools appear in no more than eight papers. This indicates, in the literature, that there is no clear, indisputable evidence or direction about which tools to employ in railway systems development, and many tools may be adequate for the same purpose.

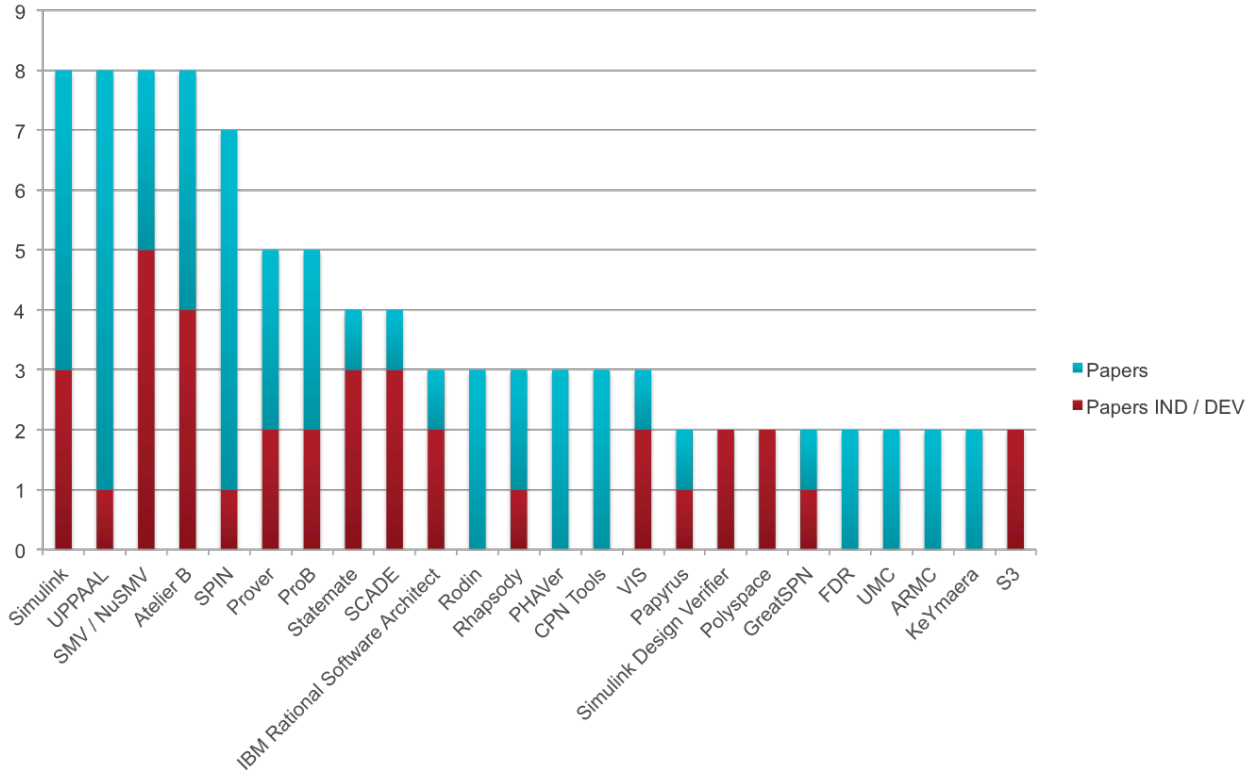


Figure 9 Tools

3.3.7 Phases

Figure 10 reports the results in terms of number of papers that address a specific CENELEC phase. Most papers address more than one phase at the same time but typically the great majority of the papers focus on the Architecture and Design phase[6]. This result indicates that evidence on the usage of formal methods is higher for the modelling and verification of the high-level definition of the control logic of a system, while there is less evidence for the subsequent phases. Indeed, Testing[7], Requirements and Component Design phases are considered, but in a much smaller number of works. The statistics for industrial papers and the other papers are somehow uniform, as shown in Figure 10, but it should be noticed that industrial works tend to cover both the Requirements phase, and later phases of development such as Validation, Implementation and Maintenance. This suggests that in industry there is an interest to cover more phases of development with formal methods, and there is a greater interest for phases of development that are closer to the

[6] We considered a paper to address this phase also when formal verification was applied to the high-level model of a system. Indeed, although formal verification may be seen as a means to verify the system, and therefore as a potential replacement for testing, this is not true for the inspected works: the code still needs to be tested, since the code is normally at a lower level of abstraction with respect to the models available in the Architecture and Design phase.

[7] We considered a paper to address this phase when the actual code was tested. Hence, papers using model-based testing for testing a model of the system are not considered to address the Testing phase.

implementation level, in which formal methods can be used to guarantee the correctness of the actual implementation, either by means of testing or by means of verification of lower level components.

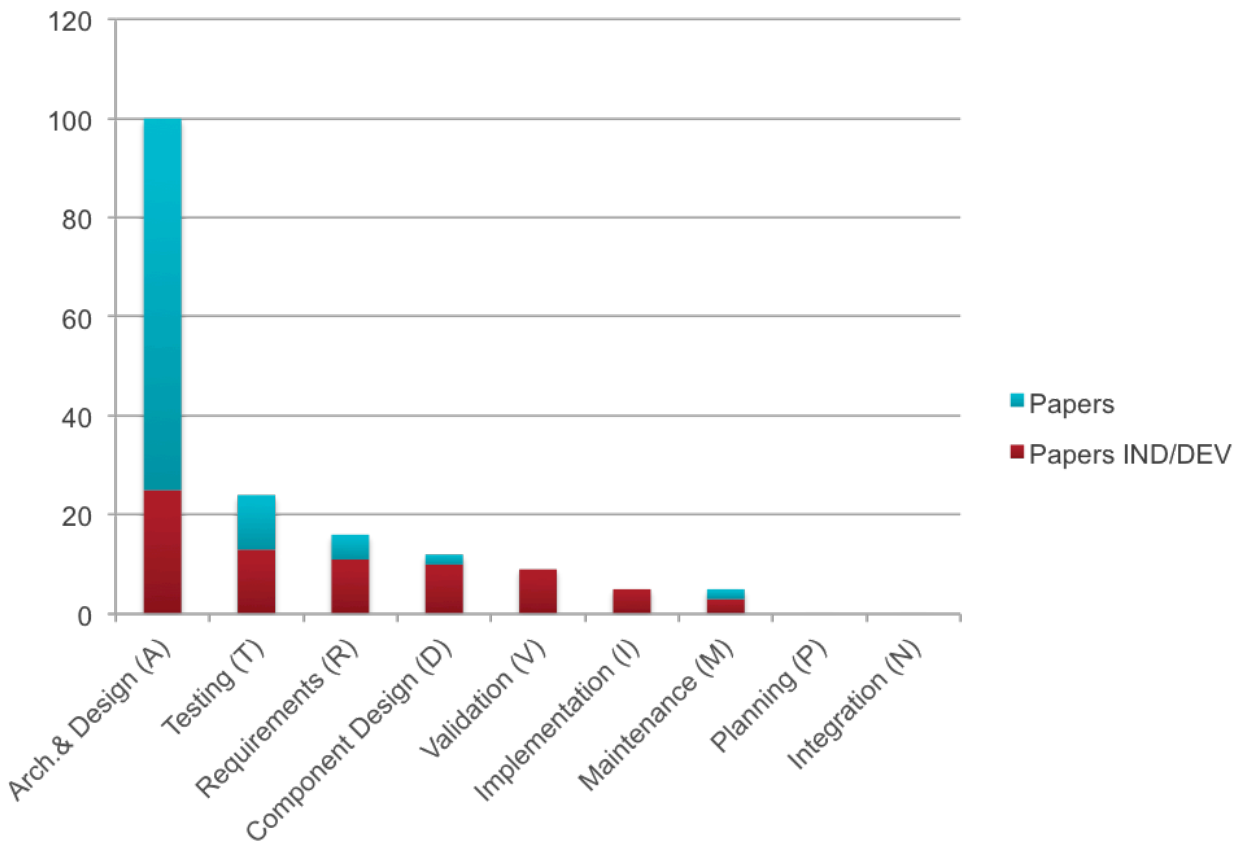


Figure 10 Phases

3.3.8 Tasks

Figure 11 reports the results in terms of number of papers that address a specific task. Specification / Modelling and Formal Verification are dominant, indicating that these tasks are the ones for which there is higher maturity in the railway field. Also model-based testing and refinement are addressed, while less works focus on the actual source code, with few contributions on code generation and static analysis. The results of industrial papers are in line with those of other papers. Based on our analysis, we argue that the tasks that are closer to implementation are less mature in the formal methods literature, and require more research. It should be noticed that, from the previous section, we have suggested a possible interest in industry towards later stages of development, but more works are required to provide evidence on the application of formal methods for code generation and static analysis in the railway field. Nevertheless, these results can provide guidance for the evaluation of formal tools for railways among the tools review presented in Section 6.

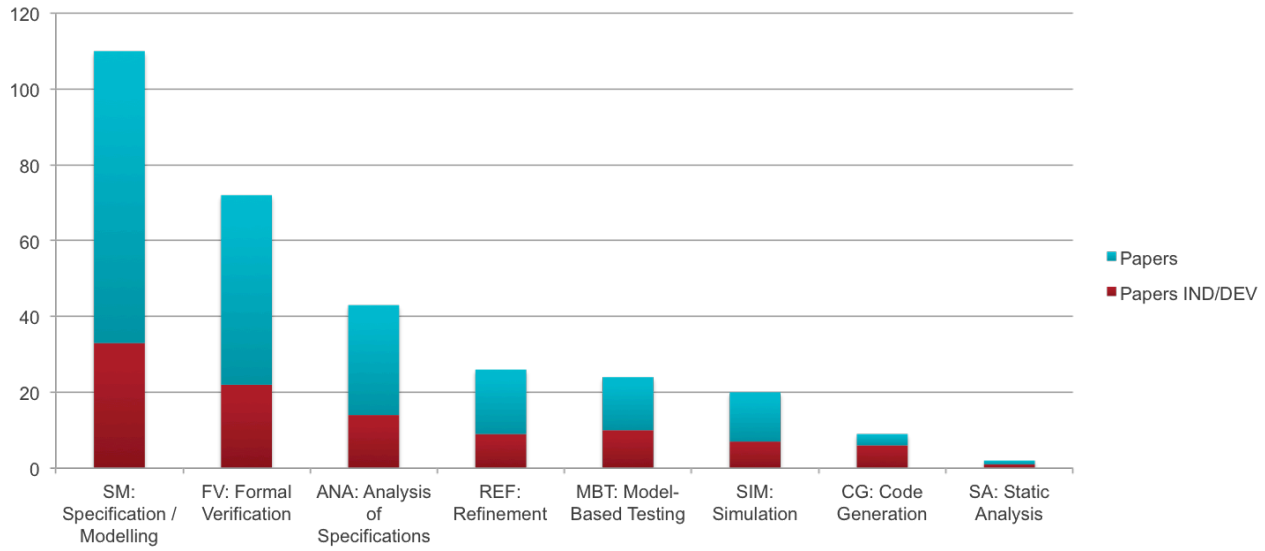


Figure 11 Tasks

3.3.9 Categories and Subcategories of Railway Systems

Figure 12 reports the results in terms of number of papers that present the application of formal/semi-formal methods to a specific type of system. The majority of the works are dedicated to Stand-alone systems and Non-standard systems. However, a representative number of papers focus on ERTMS / ETCS systems and CBTC ones. Less relevant are the contributions to the CTCS systems, and none of the works on CTCS are industrial ones. Although these statistics do not provide much guidance *per-se* on the usage of formal methods for the development of railway systems, the reader interested in developing a system according to certain standard, as, e.g., CBTC or ERTMS / ETCS, can access the papers listed in Annex 1 to identify the actual works that have presented a development according to these standards. The papers can be used as an actual guidance on the application of formal methods to these systems' standards.

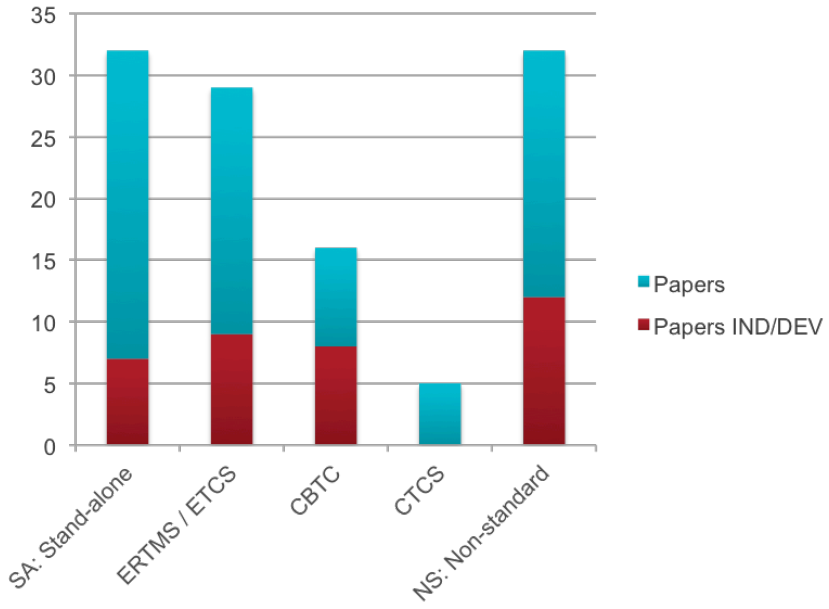


Figure 12 Categories

Figure 13 focuses on the subsystems developed with the support of formal/semi-formal methods. We recall that our review does not include interlocking products, but focuses on all the other types of railway systems. We see that a large spectrum of subsystems is covered by the different works. The majority of the works focus on High-level Control Logic and Communication, presenting the modelling and verification of the interaction between different subsystems, modelled at a high-level of abstraction. Also Railway Crossing controllers and ATPs are frequently subject of the different studies, with a major industrial evidence for ATP systems with respect to Railway Crossing controllers, which are often used as academic examples to demonstrate a certain technique. The ERTMS / ETCS Radio Block Centre, Door Controllers, ATO and ATS follow these systems in terms of number of related papers.

As for the categories of railway systems, the interested reader can refer Annex 1, to identify the papers that develop a certain system using formal methods.

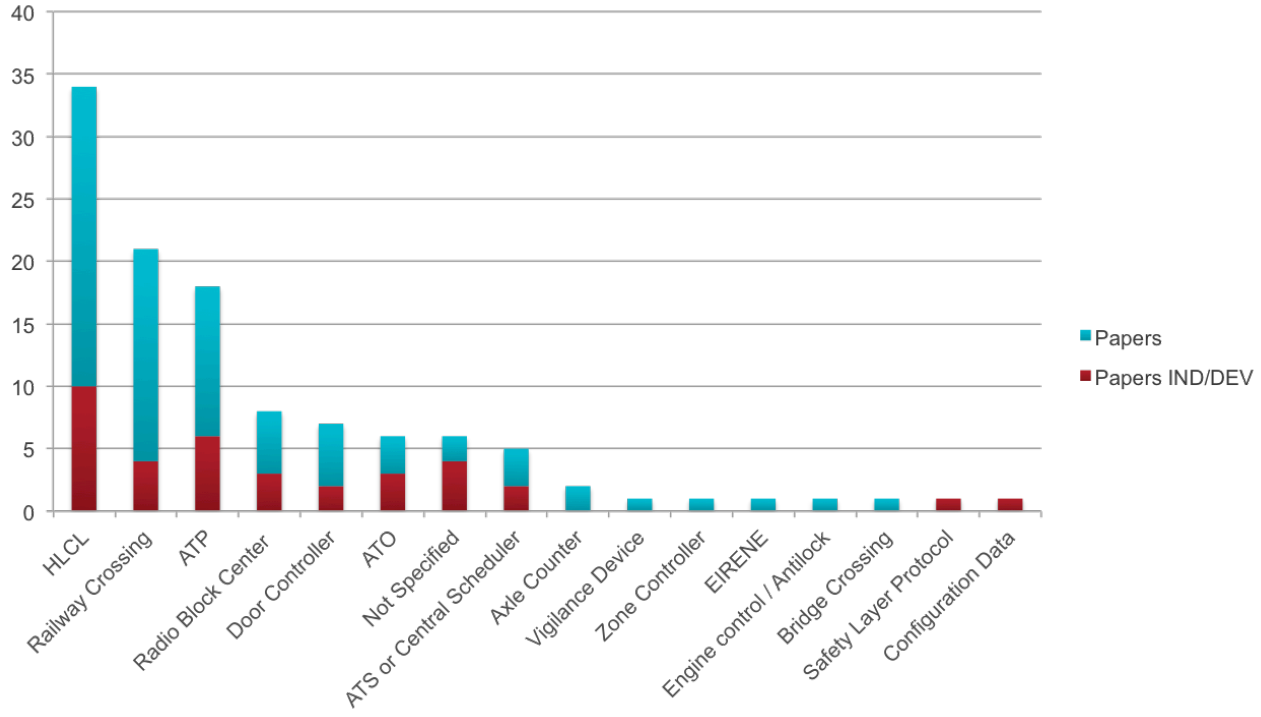


Figure 13 Sub-categories

3.4 Summary and Discussion

This section provides a discussion on the results, by providing an answer to the different research questions based on the results reported in the previous sections.

3.4.1 RQ1: What primary studies have been conducted about applications of formal and semi-formal methods in the railway domain?

As highlighted in Section 3.3.3 the studies analysed are mainly on rigorous Formal methods, while more limited are the studies on Semi-formal methods. In industrial papers, it was observed a more pronounced tendency to combine Formal and Semi-formal methods. Therefore, based on literature evidence, we argue that, in industrial railway applications, it is common to use Semi-formal methods to enable high-level reasoning on the system by modelling it and possibly simulating its behaviour, and then to pass to the usage of Formal methods for the formal verification of properties.

3.4.1.1 RQ1.1: Which types of methods are used?

According to Section 3.3.4, the most used techniques are model-based development, model-checking, and theorem proving. Industrial works show that model-based testing, a sub-category of model-based development, is also a technique of interest. From Sections 3.3.5 and 3.3.6 we see that a wide, fragmented variety of input languages and tools is used in the reviewed literature. From Section 3.3.5 we see that the most used language is UML, a semi-formal language for system modelling, State Machines in their different dialects and the B / Event B language. Furthermore, from Section 3.3.6, the most used tools appear to be the ones for the B method, such as Atelier B and ProB, followed by Simulink, UPPAAL, NuSMV, SPIN and Prover. It should be noticed that, in the papers using the UML language, the name of the tool adopted for

UML modelling was normally not mentioned. Based on this evidence, we argue that, in the railway applications presented in the papers, it is common to use UML as the semi-formal language to use for high-level modelling. Then, depending on the degree of formality required and the case at hand, the papers use either the B Method, NuSMV, SPIN, UPPAAL, or Prover, in case high formality is required, or Simulink, in case lower formality is required.

3.4.1.2 *RQ1.2: In which phases of the system development are the methods applied?*

According to Sections 3.3.7 and 3.3.8, formal methods are mainly used in the Architecture and Design phase, for modelling and formal verification of systems. These phases and activities are therefore the ones for which there is sufficient evidence of formal methods application. Later development phases and tasks related to the analysis or testing of code received less attention according to the literature.

3.4.1.3 *RQ1.3: To which type of railway systems are the methods applied?*

According to Section 3.3.9 most of the works focus on the high-level modelling and verification of the interaction and communication between different railway subsystems. Several works also focus on Railway Crossing controllers, ATP, ERTMS / ETCS Radio Block Center, ATO and Door Controllers. Reference railway and metro standards, such as ERTMS / ETCS and CBTC, are largely considered in the literature, although most of the work focus on non-standard systems.

3.4.2 RQ2: Which methods are the most mature for industrial application?

To answer this RQ, we consider the papers that are marked as IND/DEV, which indicate works with industrial participation. We recall that the answer to this question is given for the railway context, and for non-interlocking systems. According to Sections 3.3.5 and 3.3.6 if we consider solely the tools and languages used in industrial works, the most mature languages appear to be State Machines / Statecharts, UML and B / Event B. The literature appears to show an acceptable amount of evidence in this sense, with more than five industrial scientific publications for each language. Furthermore, non-industrial works also confirms the dominance of these languages. Less evidence is available for tools. If we arguably consider a tool to be industrially mature if it is used in at least two industrial works, the tools that can be considered mature are: **Simulink[8], NuSMV, Atelier B, Prover, ProB, SCADE, IBM Rational Software Architect, Polyspace, S3**. Statemate also appears to be mature, but there is no recent work using the tool, and the tool appears not to be maintained anymore by IBM[9]. A similar situation occurs for VIS, which does not appear to be used in recent publications, and does not appear to be currently maintained.

As mentioned, these considerations on tools are based on fragmented evidence from the literature, and no empirically grounded answer can be given on the most appropriate tools to employ for railway software development. However, in the context of this project, this information is considered sufficient to be used as first guidance for selecting relevant tools to be evaluated during the tool review. It should also be noticed that these conclusions are applicable solely based on the published evidence, and do not take into account possible experience performed in industry with formal tools, if they do not have any associated scientific

[8] Simulink Design Verifier, used in two publications, is a Simulink package; therefore we implicitly include it in this list by mentioning Simulink.

[9] https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=897/ENUS5724-X72&infotype=OC&subtype=NA&appname=skmwww

publication. To have an insight on tools that may be neglected by the literature, we complement the SLR with a Projects Review and a Survey with railway practitioners, which are presented in the following sections.

4 Projects Review

4.1 Methodology for Projects Review

The Projects review has been based on the search for projects that in the last twenty years have addressed the use of formal methods and tools in railways applications. The available documentation for each project has been examined in order to list the formal methods used or recommended, both as languages and as tools, in reference to the phases and tasks listed in section 3.2.5 of the Systematic Literature Review.

4.2 Results from the Projects Review

We have found, from available documentation like papers and web pages, 14 projects that starting from 1998 to today have addressed the use of formal methods and tools in railways applications. The projects are reported in the table below, where they have been divided according to the different types of railway system considered: ERTMS/ETCS/CBTC, distributed railway signalling and interlocking.

	ERTMS/ETCS/CBTC
CRYSTAL	http://www.crystal-artemis.eu/
Deploy	http://www.deploy-project.eu/
DITTO	http://cs.swansea.ac.uk/dittorailway/
EuRailCheck	https://es.fbk.eu/projects/eurailcheck-era-formalization-and-validation-etcs
MBAT	http://www.mbat-artemis.eu/home/69-abstract.html
OpenCOSS	http://www.opencoss-project.eu
OpenETCS-ITEA2	http://openetcs.org/
PERFECT	https://trimis.ec.europa.eu/project/performing-enhanced-rail-formal-engineering-constraints-traceability
	Distributed railway signalling

SafeCap	http://gow.epsrc.ac.uk/NGBOViewGrant.aspx?GrantRef=EP/I010807/1
	Interlocking
ADVANCE	http://www.advance-ict.eu/
EULYNX	https://eulynx.eu/
EuroInterlocking	http://test.swissrequirementsengineering.ch/en/projects/euro-interlocking-project
INESS	http://www.iness.eu
RobustRail	http://www.robustrails.man.dtu.dk

Table 1 Projects

Similarly to what done in the Literature review, for the purpose of this deliverable we focused on the experiences done in the first bunch of projects, that is, those in which the considered railway systems are train protection systems referring to **ERTMS/ETCS** or **CBTC** standards.

The quality and heterogeneity of documentation has not allowed to fully compare quite different projects. Regarding the phases and tasks of interest their focus turns out to be mostly non requirements modelling, formal specification and on techniques and tools to analyse verify and validate the specifications.

Figure 14 shows the adopted modelling languages while Figure 15 show the tools used.

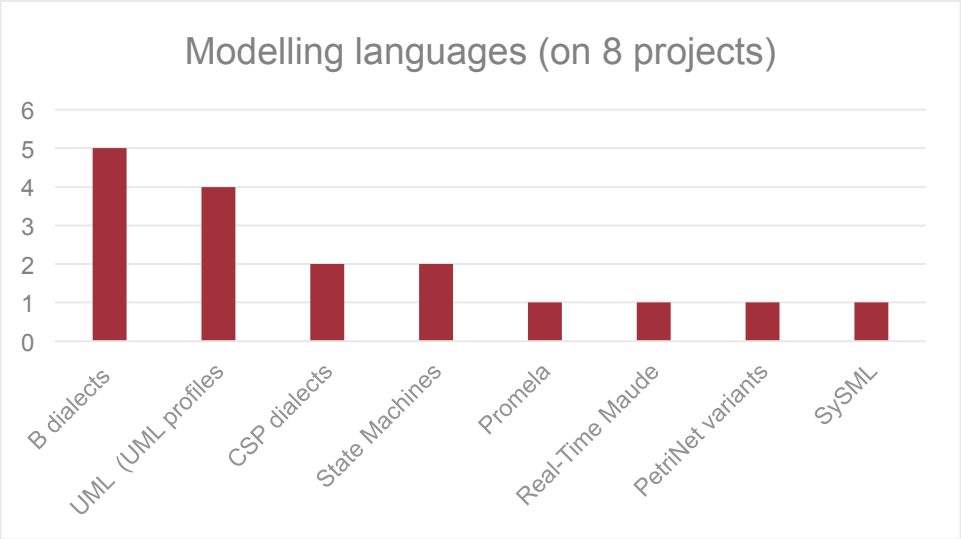


Figure 14 Modelling Languages from Projects

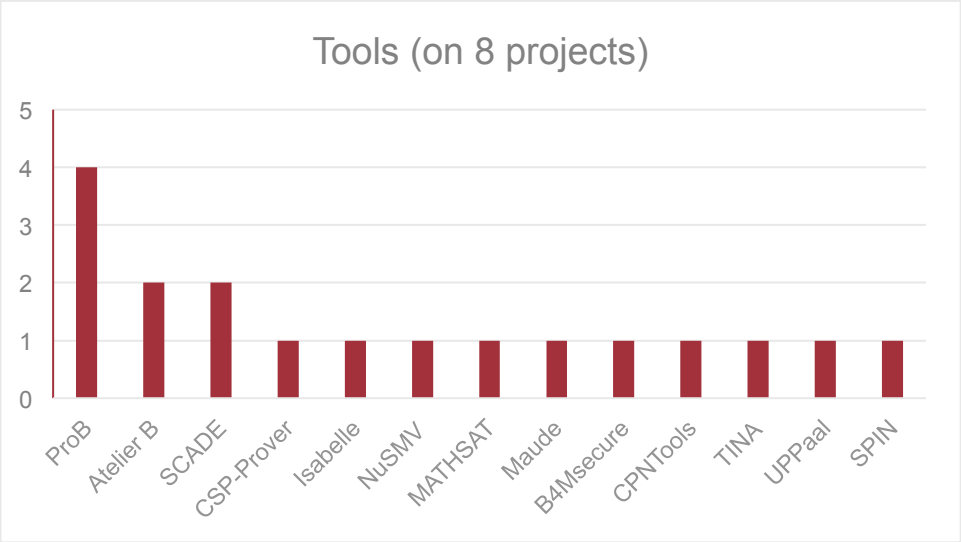


Figure 15 Tools from Projects

The two figures substantially confirm the information extracted by the Systematic Literature Review, with a prominence of the “B eco-system”, but otherwise confirming the industrial preference to UML/SysML as modelling languages, followed by different state machines languages, and the importance of a commercial tool as SCADE, emerging from a number of academic tools, mostly dedicated to formal verification.

5 Survey with Practitioners

A survey with railway practitioners in the form of a questionnaire was conducted to (a) assess that the information elicited through the SLR about the uptake of formal methods in railways was consistent with the viewpoint of practitioners, and (b) to evaluate the relevance of the different evaluation criteria adopted for

evaluating the different formal tools. In the following, we describe how the questionnaire was designed, and then we discuss the results.

5.1 Questions Definition

A questionnaire with stakeholders has been carried out by means of a questionnaire, submitted to the participants of the recent RSSRail'17 conference[10]. The venue is normally attended by academics and practitioners interested in the application of formal methods in railways, and was therefore considered as the source of a population sample that could provide an informed judgment on the topic[11].

The goal of the questionnaire was to:

a) identify the current uptake of formal and semi-formal methods and tools in the railway sector;

b) identify the features, in terms of functional and quality aspects, that were considered more relevant for the application of a certain formal tool in the development of railway products.

The questionnaire was designed to be easily understood by the target group, involving academics and practitioners, and to be filled within five minutes, to limit the amount of time required to the people surveyed, and possibly increase the number of respondents. The design of the questionnaire was performed by the authors of the current paper, and was tested and validated with industrial partners of the ASTRail consortium for clarity and time required. An online version of the questionnaire, which the reader can refer to have a clear view of the proposed questions, can be found at the following link: https://goo.gl/forms/4b9wSTJAMOK7VghW2.

In the following sections, we report the results obtained, and the associated observations.

5.2 Questionnaire Results

5.2.1 Affiliation and Experience

The first part of the questionnaire was dedicated to identify the respondents in terms of affiliation and experience in railways and formal/semi-formal methods and tools. A total of 44 subjects responded to the questionnaire, and the respondents are balanced between academics (50%) and practitioners (50%–47.7% from railway companies, and 2.3% from aerospace and defence). Figure 16 shows the experience of the respondents in railways and in formal methods, respectively, in terms of years. We see that the majority of the respondents has more than ten years of experience in railways (38.6%) and in formal methods (52.3%), and this confirms that our sample can provide informed opinions to the remaining questions.

[10] https://conferences.ncl.ac.uk/rssrail/

[11] We do not have an estimate of the whole population of subjects applying formal methods in railways. However, assuming that the population of subjects applying formal methods in railways is 1,000, our results on a sample of 44 subjects are valid for a confidence level of 85% and margin of error of 10.5%—higher values are normally targeted in qualitative research; however the answers to the questionnaire, presented later, show that the sample is made of high-quality (i.e. informed) respondents, which increases the reliability of our results.

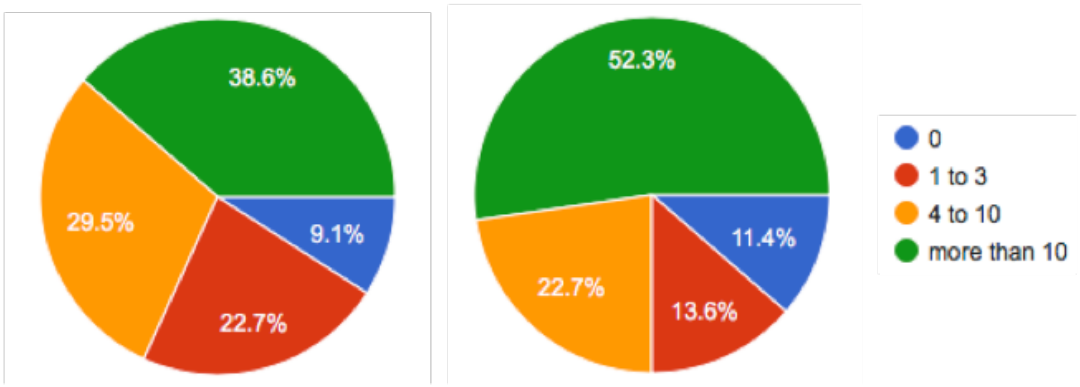


Figure 16 Experience in Railways (left) and in Formal Methods and Tools (right)

5.2.2 Usage of Formal Methods in Railways

The second part of the questionnaire was oriented to have an insight on the usage of formal/semi-formal methods and tools in railways.

Projects We asked in how many industrial railway projects the respondents, or their teams, used formal/semi-formal methods and tools. Since the respondents included also academics, we expect that the industrial projects in which they were involved were mainly technology transfer projects with companies. Figure 17 shows that only three of the respondents — or their teams — (6.8%) did not have any industrial experience in the application of formal methods in railways. The reader will notice that this number is lower with respect to the ones who declared to have zero years of experience in railways, or formal methods (cf. Figure 16). Indeed, in this question, we explicitly asked whether the respondents, or their teams, applied formal tools to railway problems, while the previous question was directed to the respondent only. Therefore, a respondent, e.g. without experience in railways, may belong to a team that applied formal methods to a railway product.

It is particularly interesting to compare these numbers with the statistics about the years of experience reported in Figure 16. While most of the respondents have more than ten years of experience in railways and formal methods, the majority of them applied formal methods in less than ten projects. This confirms a typical characteristic of railway projects, which normally last far more than one year, due to the complexity of the products, and length of the safety-critical process that the development of railway products needs to follow.

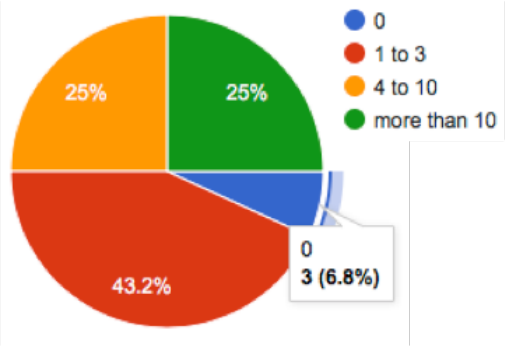


Figure 17 Number of projects on formal methods in railways

Phases Figure 18 shows the phase of the development process in which formal methods are applied. We see that all the phases have been selected by at least one of the respondents, highlighting the potential pervasiveness of formal methods within the railway process. Most of the respondents (73.8%) used them for specification, and for formal verification. Analysis of specifications (50%) and simulation (10.5%) appear also to be common, and a non-negligible amount of respondents (31%) used formal methods also within model-based testing and code generation contexts. Less common (7.1%) is the application of formal methods for the static analysis of the source code.

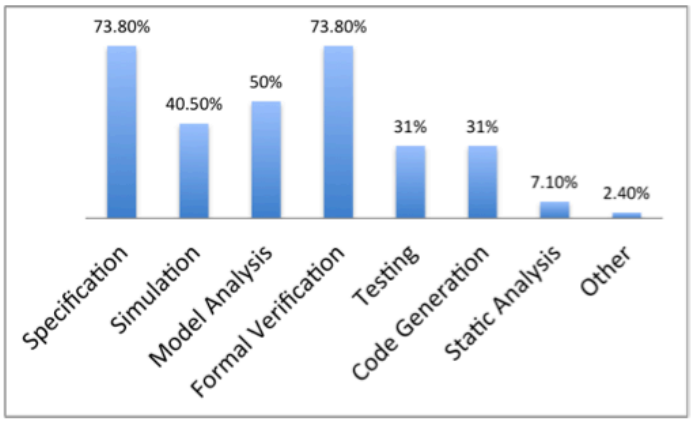


Figure 18 Phases in which formal methods are applied

Tools The respondents also listed the tools used in the context of their projects. From Figure 19, we can see that the large majority of industrial and academic respondents mentioned tools belonging to the B-method family (e.g. B, ProB, AtelierB, EventB, RODIN). The relationship between the B-method and the railway sector is well established, as well known. Other methods and tools mentioned by both groups are the Matlab toolsuite — including Simulink and Stateflow — SCADE, Petri nets/CPN tools NuSMV and SPIN. A large list of other tools is mentioned, with only one respondent per tool. Hence, we do not consider these tools in the statistics.

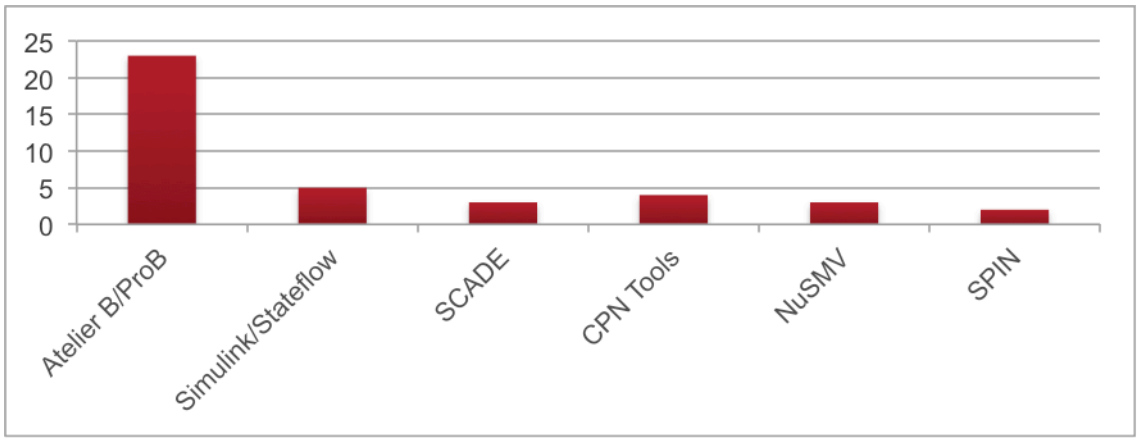


Figure 19 Tools used in the context of the projects

5.2.3 Relevance of Evaluation Criteria for Formal Tools

The questionnaire is used to establish the relevance of the different evaluation criteria to be used in the ranking function for the ASTRail project. Functional Aspects and Quality Aspects are evaluated. Language specific aspects are not considered in the questionnaire, since they are considered as too much technical, and project/phase dependent to be evaluated in the context of the questionnaire.

The final part of the questionnaire was dedicated to identify the most relevant features that a formal/ semi-formal tool should have to be used in the railway industry. Features are partitioned into functional and quality aspects. We asked to check at most three relevant functional features, among the seven listed, and at most five relevant quality aspects, among the sixteen listed.

Functional aspects Figure 20 shows the results for the most relevant functional features. All the listed features are considered relevant by at least one of the respondents. The functional features that are considered most relevant by the majority of the respondents are formal verification (86.4% of the respondents), followed by modelling—graphical or textual—(72.7%). These traditional functional features of formal tools are followed by simulation (30%) and traceability (27.3%). Indeed, simulation (often in the form of animation of a graphical specification) is needed for a quick check of the behaviour of a model; traceability between the artefacts of the software development (requirements to/from models, models to/from code, etc.) is mandatorily required by the main guidelines for the development of safety critical systems. Functional features, such as test generation and code generation, related to later activities of the development process, are also considered relevant by a non negligible amount of respondents (22.7%). These numbers suggest that formal tools are seen to play a role mostly in the early phases of the development process, for specification and formal verification. These are also the phases for which formal methods cannot be substituted by any other means—as it happens for testing, code development and tracing.

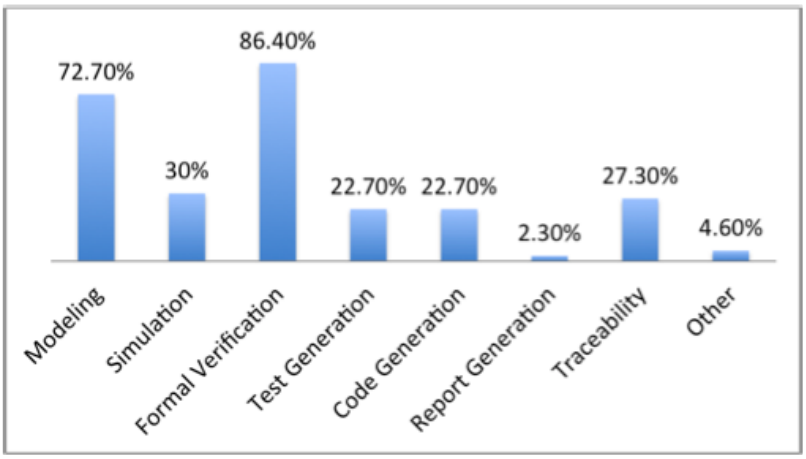


Figure 20 Most relevant functional aspects

Quality aspects Figure 21 reports the most relevant quality aspects and, also in this case, all the listed answers were checked by at least one of the respondents. The maturity of the tool (stability and industry readiness) is considered among the most relevant quality aspects by 75% of the respondents, followed by learnability by a railway software developer (45.5%), quality of documentation (43.2%) and ease of integration in the CENELEC process (36.4%). Overall, the most relevant quality aspects are associated to the usability of the tool.

Less relevant are deployment aspects, such as platforms supported (9.1%) and flexible license management (11.4%). Interestingly, also the low cost of the tool (13.6%) is not a strictly relevant feature. This is a reasonable finding. Indeed, the development and certification cost of railway products is high, and, if a company expects to reduce these costs through a formal tool, it can tolerate the investment on the tool.

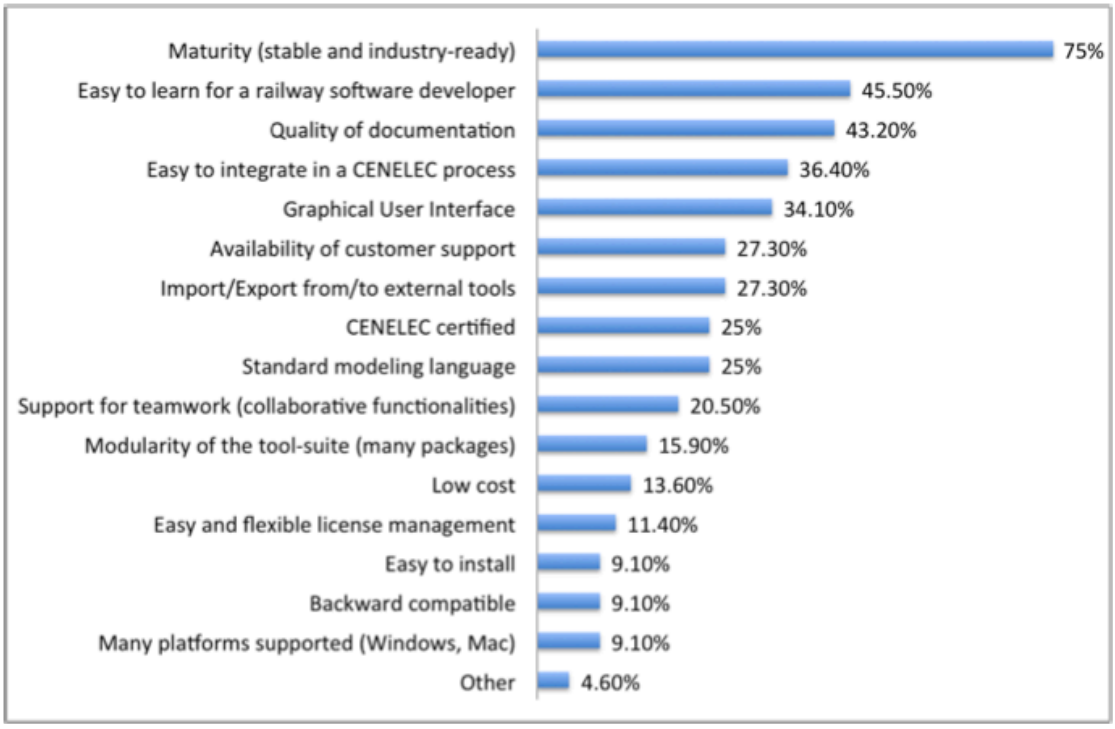


Figure 21 Most relevant quality aspects

6 Tools Review

In this section, the evaluation of the tools emerged from the previous phase of the project will be detailed. Firstly, the DESMET methodology for evaluating tools will be briefly introduced, then the evaluation process and criteria will be discussed and finally the results will be provided.

6.1 DESMET Methodology for Tools Review

DESMET[12] [KLL97, MBK14] is an evaluation methodology that has been proposed to evaluate/rank methods or tools. DESMET is composed of nine different types of evaluations and a set of criteria that are used to guide the evaluators in choosing the best evaluation, depending on their needs and those of their organizations.
The main idea of DESMET methodology is to help ranking tools based on specific information retrieved according to a particular context in which the tools have to be evaluated. In our case, the context is the *railway domain*.
In the DESMET methodology, several evaluation types are possible. It is then paramount to choose the most suitable on. The types of evaluation available can be categorised according to aspects of tools that are to be examined. The two main methods available are *quantitative* and *qualitative* studies:

- *Quantitative:* one of such methods can be selected if the primary aspects of a tool to be evaluated are the quantitatively measurable effects that the tool may have within an organisation.
- *Qualitative*: these methods are more appropriate when the objective of the evaluation is concerned with the suitability of a tool in a given setting.

[12] DESMET stands for “a methoD for Evaluating Software engineering MEthods and Tools”.

Quantitative and qualitative evaluation methods are further organised as a *formal experiment*, *case study* or *survey*. Qualitative evaluations can also be organised as a *feature analysis* ([KLL97]).
The main differences among them are:

- in a *formal experiment* many subjects are asked to perform a variety of tasks using the different tools under analysis. Subjects should be properly assigned to each tool to reduce bias. Data should be gathered through statistical techniques;
- in a *case study* each tool is tried out on a real project and by using the development cycle and procedures of the evaluating organisation;
- a *survey* is based on gathering information about tools that have been used by the organisations on past projects;
- finally, a *feature analysis* is done by a single individual (or cohesive group) who not only determines the features to be assessed and their rating scale but also does the assessment. For initial screening, the evaluations are usually based on literature describing the software method/tools rather than actual use of the methods/tools.

According to the above classification the selected method for conducting this tool review is *qualitative feature analysis*. However, with respect to a standard feature analysis (where it is only required to check the documentation) the reviewers have also tried out all analysed tools on a specific railway model. Moreover, some information used for carrying out the ranking has been retrieved by meeting with the industrial partners, with the help of questionnaires.
This type of review has been considered the most suitable one because it better fits into the needs of the ASTRail project, which roughly are a large number of tools to assess in relatively short time. In particular, the features selected for the screening will be called *evaluation criteria*, and have been selected following meeting with the industrial partners of the ASTRail project. Concerning the selection of tools for the feature analysis, these have been retrieved from the Systematic Literature Review process previously described.

6.2 The Evaluation Process

One of the questions to which the conducted SLR has provided a response is the question:

RQ2: Which methods are the most mature for industrial application? (See Section 3.4.2)

The response to that question has been shown to be the following list:

Simulink, NuSMV, Atelier B, Prover, ProB, SCADE, IBM Rational Software Architect, Polyspace, S3

Of the above list we discarded the *IBM Rational Software Architect* because it is just a design tool that does not allow any kind of formal verification, the *Polyspace* tool because it is a static analysis tool that does not support any kind of behavioural verification, and the tools *Prover* and *S3* because we neither have been able to find sufficient documentation on them, nor we have been able to access a demo version of them.
Therefore from the above list, the subset of tools on which have been selected for further more specific evaluation are **Simulink, NuSMV, Atelier B, ProB, SCADE**.

Moreover, the results of the Survey with Practitioners (as shown in Section 5.2.2 and in Figure 19 Tools used in the context of the projectsFigure 19) indicate two additional tools sometimes used in railway-related industrial projects, namely **SPIN** and **CPN Tools**. Therefore, these tools have been also selected for further specific evaluation.

Additionally, we are aware of other relevant tools and frameworks used in industrial projects, even if not widely used within the railway sector so far. Therefore, we have decided to extend our specific evaluations also to them, adding to our list the **UPPAAL**, **FDR4**, **CADP**, **mCRL2**, **SAL**, and **TLA+** frameworks.

Finally, we have also taken into consideration a last tool, namely **UMC** that, even if lacking a solid background in terms of industrial usages, has the uncommon feature of allowing a direct verification of UML based models. We recall that, according to the SLR, UML is the most common semi-formal language used for the high-level specification of railway systems.

In conclusion the final list of **14 tools/frameworks** selected for a deeper evaluation is the following:

Simulink, nuXmv[13]**, Atelier B, ProB, SCADE, SPIN, CPN Tools, UPPAAL**, **FDR4**, **CADP**, **mCRL2**, **SAL**, **TLA+** and **UMC**.

All these tools and frameworks, with the exception of SCADE, have been actually downloaded, installed, and experimented with the design and verification of one or two railway-related simple cases studies. The corresponding available tool documentation has been analysed with the depth allowed by the project timeline.

In the case of SCADE, because of licensing issues, we did not have the possibility of a "hands-on" experience, therefore our evaluation is limited to the analysis of the available official tool documentation and presentations, and to some student experiences from the University of Florence, reported in their theses making use of the tool under an University Teaching license.

6.3 Evaluation Criteria

A set of evaluation criteria is defined to evaluate the different formal and semi-formal methods. Since a method can hardly be separated from a tool, these criteria are expected to be evaluated on tools, in the context of this project.

The list of selected evaluation criteria to be used for formal methods and tools evaluation has been derived in accordance with the industrial partners of the ASTRail project following the methodology described below. The criteria were elicited with a collaborative approach inspired by the KJ method [Scu97]. Specifically, a workshop involving eight WP participants from CNR and SIRTI was organised. Each participant was given five minutes to think about relevant criteria that should be considered when evaluating a formal or semi-formal method. The participants wrote the criteria that they considered relevant in a sheet. Then, the moderator asked each participant to list the criteria written in the sheet and to briefly discuss each criterion. When the explanation given by the criterion proponent was not clear, the other participants could ask additional questions to clarify the meaning of the criterion. The moderator reported each criterion in a list. The list was edited on a shared document that was visible to all participants. When a criterion was already mentioned by another participant, the criterion itself was not added to the list. At the end of the meeting, two appointed members of the group homogenised the criteria and the final set was validated by the other participants.

In this process, the criteria have been divided into three main categories: those related to **functional aspects**, those related to **quality aspects** and those related to the **language** used. Each category contains a group of criteria entailing similar aspects. In the following, the categories above will be detailed, and the groups of criteria will be discussed.

In an industrial setting a methodology is generally identified with the tools that support it. To avoid cumbersome repetitions the following criteria may relate to a specific methodology or its corresponding tools. Moreover, the set of criteria emerged from the specific context of the project, although representative for our scope, may be extended to comply with the peculiarities of other contexts.

[13] nuXmv is the successor of NuSMV

6.3.1 Functional Aspects

The criteria concerning functional aspects have been organised in two groups: verification and development features.

Verification features This group of features is used to evaluate specific technical aspects of each tool, related to the possibility of validating the models according to different techniques.

- **Simulation** The model should be executable, so that simulation is possible. The supported simulation (if any) could be either graphical (GRA) or textual (TEX), or a mix of the two (MIX).
- **Formal verification** The model can be formally verified according to different verification techniques (e.g. linear time model checking (MC-L), branching time model checking (MC-B), model checking of state invariants (MC-I), theorem proving (TP), refinement checking (RF).
- **Supported problem size** The indicative maximum size, in terms of reachable states, that the tool is able to handle for a complete analysis (e.g. less than a few million states (LIMITED), few hundreds of millions (MEDIUM), billions or more (LARGE).
- **Model-based testing** The tool should provide automatically derived testing scenarios. The methodology should generate coverage tests (YES, NO).
- **Time related aspects** The tool should allow the verification of time related properties (YES, NO).
- **Probability aspects** The tool should allow the verification of probability related aspects (YES, NO).
- **Property specification language** The names of the languages supported for the specification of properties (this aspect is just informative).

Development features This group of features specifies aspects related to the development of models and code with the specific tool, and if requirements can be managed and documents produced.

- **Specification, modelling** It evaluates whether the model can be edited graphically (GRA) or in some textual representation (TEXT) inside the framework, or whether the model is just imported as a textual file (TEXTIN).
- **Code generation** It indicates whether the tool supports automated code generation from specifications (YES, NO).
- **Documentation and report generation** It evaluates the possibility of automatically generation of documents, and their quality. Those documents shall be clear enough to be used as official documents with minor editing (YES, PARTIAL, NO).
- **Support for requirements traceability** It should be possible to trace requirements from specification down to code implementation (YES, NO).
- **Model refinement** It should be possible to design the system as a hierarchy of models at different levels of abstraction (YES, NO).

6.3.2 Language Aspects

Language expressiveness This group of features entails technical aspects related to the modelling language available in the tool.

- **Name of language** The name of the language used by the tool (this aspect is just informative).
- **Non-determinism** It evaluates if and how non-determinism is expressible. In particular whether the models allows non-deterministic system evolutions (INT), whether the model supports non-determinism associated to external inputs or trigger-events (EXT), or the model is fully deterministic (NO).
- **Concurrency** It evaluates if and how concurrency aspects can be modelled through its formalism. In particular whether the model can be constituted by a set of asynchronously interacting elements

(ASYNCH), whether the model can be constituted by a set of synchronous elements (SYNCH), or whether the model is constituted by just one element (NO).

- **Temporal aspects** It evaluates if temporal aspects can be modelled (e.g. timed automata) (YES, NO).
- **Stochastic aspects** It evaluates if it is possible to model probabilistic aspects, as for example stochastic delays (YES, NO).
- **Modularity of the language** It evaluates how the architecture of the model can be structured in the form of different hierarchically linked modules. In particular, whether the tool allows the user to model in a hierarchical way, and the partitioning of the model into modules (HIGH), whether the tool allows the partitioning of the model into modules but does not allow the user to model in a hierarchical way (MEDIUM), or whether the tool allows the partitioning of models into modules, but the modules have no way to interact, neither by messages nor by shared memory (LOW).
- **Supported data structures** It evaluates how different data structures are supported. In particular whether language supports just elementary types like Boolean and integers (MINIMAL), whether the language supports various kind of numeric types, but no composite expressions (BASIC), or whether the language supports complex expressions like sequences, sets, array values (COMPLEX).
- **Model kind** The approach used to specify the models. The choice is among an Imperative style, a Functional style, an Algebraic style, a Logical style, or a Graphical approach (this aspect is just informative).

6.3.3 Quality Aspects

The features concerning quality aspects have been organised into six groups: maturity, usability, company constraints, domain-specific criteria, methodology flexibility and language expressiveness.

Maturity This group of features indicates whether the tool is prototypical, or it is established, possibly with several versions across the years.

- **Industrial diffusion** This feature aims to evaluate whether the methodology is widely used in railway applications and specifically oriented to railway system (RAILWAY) or whether the methodology is sometimes used for railway system even if the tool is not specifically oriented for this kind of systems (MEDIUM), or whether it is only partially used in industry without evidence of uses in the railway sector (LOW).
- **Stage of development** This feature aims to evaluate whether the tool is a stable product with a long history of versions (YES), whether the tool is a recent tool but with a solid infrastructure (PARTIAL), or whether the tool is a prototype (NO).

Usability This group collects the features concerning usability aspects.

- **Availability of customer support** It is evaluated whether reliable customer support can be acquired for maintenance and training (YES), or free support is available for maintenance and training (PARTIAL), or communications channels are established among producers and users (NO).
- **Graphical user interface** It is evaluated whether the user interface is graphical or by command-line. E.g. the tool has a well defined and powerful graphical user interface (YES), or a user friendly GUI exists, but does not cover all the tool functionalities in a graphical form (PARTIAL), a GUI exists, but not particularly effective in the design and usability (LIMITED), the tools is a command line tool (NO).
- **Easy to use** It evaluates the learning curve for the specific tool. In particular we classify the kind of mathematical /logical background needed for an effective use of the tool.
- **Quality of documentation** It is evaluated whether the documentation is extensive, updated and clear, includes examples that can be used by domain experts, it is accessible and navigable in an easy way (EXCELLENT), whether the documentation is complete but offline, and requires some

effort to be navigated (GOOD), or whether the documentation is not sufficient or not easily accessible (LIMITED).

Company constraints This set of features entails those aspects related to the adoption of a specific methodology from a company, as for example financial or management aspects.

- **Cost for industrial usage** It is used to evaluate the costs of different available versions of the tools for the specific methodology. In particular whether the tool is available under payment only (PAY), free under limited conditions (academic) and at moderate cost for industrial uses (MODERATE), or whether the tool is freely usable for all industrial or academic uses (ZERO).
- **Supported platforms** It is used to evaluate the possible platforms supported by the tools for the specific methodology (e.g. Windows, MacOS, Linux, Solaris).
- **Easy and flexible license management** It is evaluated the complexity of the management of the licenses for the tool. In particular, whether the tool is free for commercial use, and no license management system is required (EASY), whether the tool has a free version and a commercial one (MODERATE), whether being the tool is only available upon payment, the licensing information provided in the website of the tool is clear and accurate (ADEQUATE), or whether the tool is only available upon payment, or the information provided by the website of the tool concerning license management is limited (COMPLEX).
- **Easy to install** It is evaluated whether the tool is easy to install, and require little or no dependencies to external components (YES), whether the tool installation depends on non trivial external components, or whether the installation process might not be smooth (PARTIAL), or the installation process can interfere with the customer development environment (NO).

Domain-specific criteria This group lists those features dealing with domain-specific, i.e. railway industry, criteria.

- **CENELEC certified** It evaluates the presence of a CENELEC certification, or availability of a certification kit. In particular whether the tool is certified according to the CENELEC norm (YES), whether the tool includes a CENELEC certification kit, or if the tool is certified according to other safety-related norms like DO128C (PARTIAL), or if none of the above is true (NO).
- **Easy to Integrate into a CENELEC process** It evaluates whether it is easy to integrate the methodology in the existing railway life-cycle as described by CENELEC norms. In particular it is evaluated whether the tool or language is mentioned in the text of the CENELEC norm, if in the literature and in the tool documentation was found evidence of tool usage for the development of railway products according to the CENELEC norms (YES), or whether in the literature and in the tool documentation evidence was found of the usage of the tool in railways, but any evidence was found of CENELEC products developed with the support of the tool (MEDIUM), or whether in the literature and in the tool documentation no evidence of usage of the tool in railways was found (LOW).

Flexibility This group concerns those features evaluating the flexibility of a tool: how the tool interfaces with previous versions, other tools and different phases of software life-cycle.

- **Backward compatible** It validates the degree in which models developed with previous versions can be used in the current version. In particular whether the vendor guarantees that legacy versions of the models can be used in the current version of the tool or the future availability of legacy versions of the tool (YES), whether the tool is open source, or the input language is stable and standard de facto, or there is evidence of interest in preserving backward compatibility (LIKELY). Moreover, whether the tool is not open source and the provider does not show evidence regarding the backward compatibility (even if the language is rather stable and standard de facto) (MODERATE), and finally whether sources are not available, input format is not necessarily stable, and no information available from vendor (UNCERTAIN).

- **Standard input format** It evaluates the accessibility of the input language of the tool. In particular, whether the input language is not proprietary, open and public (YES), whether the structure of the model specifications is easily accessible, but not publicly documented (PARTIAL), or whether the internal structure of the model specification is hidden (NO).
- **Import from or export to other tools** It evaluates how much the tool is able to interact with other tools. In particular, whether the tool provides several import/export functionalities (HIGH), whether the tool has a standard format used by other tools, or exports w.r.t. to other formats (MEDIUM), or whether the tool is not oriented towards import/export functionalities (LOW).
- **Modularity of the tool** It evaluates whether the methodology is not monolithic but addresses different modules and packages for the different phases of the development process. In particular, whether the tool is composed of many packages that can be loaded to address different phases of the development process (HIGH), whether the tool can address different phases of the development process in a monolithic way (MEDIUM), or whether the tool is only used in a single development phase (LOW).
- **Support for teamwork** It should support collaborative team development (YES,NO).

A reference template of tool evaluation sheet has been produced. The full collection of the various evaluation sheets are presented as a separate Annex 3 to this Deliverable. Each sheet includes, beyond the above evaluation criteria, additional informative items like the origin of the tool, the site from which it can be retrieved, the places where the most relevant pieces of information on the tool can be found, the articles that report on industrial usages of the tool.

6.4 Experimentation Results

For each of the considered tool features, the compiled evaluation sheet in Annex 3 provides results of the experimentation and evaluation giving, when useful, additional comments.

In Figure 22 and Figure 23 (7 tools per page) all the results of the evaluation are briefly summarised, using the keywords introduced in the previous section.

Figure 22 and Figure 23 also represents the base for the generation of the subsequent Ranking Matrix (Annex 2), as described in Section 7.2.

6.5 Summary and Discussion

In this section, we described the process followed for the evaluation of 14 different formal and semi-formal tools to assess their suitability for railway system development. The set was selected based on the information retrieved from the SLR, the Pojects Review and the Survey with Practitioners, which enabled the identification of the most used formal tools in the railway industry. Furthermore, the set of tools was enriched with formal tools that are widely used in domains different from railways. A set of evaluation features was defined, including functional features (e.g., formal verification, code generation), quality aspects (e.g., usability and maturity) and language-related aspects to evaluate the tool. The documentation of the tools was inspected, and a trial was performed to assess the different features.

Overall, the majority of the tools offer formal modelling and formal verification through model checking, and they generally offer simulation in textual or graphical form. Less frequent are features such as code generation, model-based testing and traceability. With few exceptions, such as SCADE and Simulink, graphical user interfaces (GUIs) for these different tools are rather limited. Furthermore, in terms of learnability, the tools mainly require medium to advanced competences in formal methods, and in the majority of the cases, require the support of an expert to be successfully used. It is also worth noticing that solely one of the tools, namely SCADE, is certified according to the CENELEC norms.

Category	Name	SPIN	Simulink	nuXmv	ProB	AtelierB	UPPAAL	SCADE
Development Functionalities	Specification/ Modeling	TEXT	GRA	TEXTIN	TEXT	TEXT	GRA	GRA
	Code Generation	NO	YES	NO	NO	YES	NO	YES
	Document / Report Generation	PARTIAL	YES	NO	PARTIAL	PARTIAL	PARTIAL	YES
	Requirements Traceability	NO	YES	NO	NO	NO	NO	YES
	Model refinement	NO	YES	NO	YES	YES	NO	YES
Verification Functionalities	Simulation	TEX	GRA	TEX	MIX	NO	GRA	GRA
	Supported problem size	LARGE	LIMITED	LARGE	MEDIUM	MEDIUM	MEDIUM	LIMITED
	Formal Verification	MC-L	MC-I	MC-L,MC-B	MC-L,MC-B, TP, RF	TP, RF	MC-L	MC-I
	Model-based Testing	NO	YES	NO	YES	NO	NO	YES
	Time related properties	NO	YES	YES	NO	NO	YES	YES
	Probability properties	NO	NO	NO	NO	NO	YES	NO
Language Expressiveness	Nondeterminism	INT	EXT	INT,EXT	INT,EXT	INT,EXT	INT	EXT
	Concurrency	ASYNCH	NO	SYNCH	NO	NO	SYNCH	SYNCH
	Temporal Aspects	NO	YES	NO	NO	NO	YES	YES
	Probability	NO	NO	NO	NO	NO	YES	NO
	Modularity of the Language	HIGH	HIGH	MEDIUM	LOW	LOW	MEDIUM	HIGH
	Supported Data Structures	BASIC	COMPLEX	BASIC	COMPLEX	COMPLEX	COMPLEX	COMPLEX
Tool Flexibility	Backward Compatibility	LIKELY	LIKELY	LIKELY	LIKELY	MODERATE	LIKELY	LIKELY
	Standard Input Format	YES	PARTIAL	YES	YES	YES	PARTIAL	PARTIAL
	Import/Export	MEDIUM	LOW	MEDIUM	HIGH	MEDIUM	LOW	LOW
	Modularity of the Tool	LOW	HIGH	LOW	LOW	MEDIUM	LOW	HIGH
	Team Support	NO	NO	NO	NO	YES	NO	NO
Maturity	Industrial Diffusion	MEDIUM	HIGH	MEDIUM	RAILWAY	RAILWAY	MEDIUM	RAILWAY
	Stage of Development	YES	YES	YES	YES	YES	YES	YES
Usability	Customer Support	PARTIAL	YES	PARTIAL	YES	YES	YES	YES
	Graphical User Interface	LIMITED	YES	NO	PARTIAL	PARTIAL	PARTIAL	YES
	Easy to Use	MEDIUM	BASIC	MEDIUM	MEDIUM	ADVANCED	MEDIUM	BASIC
	Quality of Documentation	GOOD	EXCELLENT	GOOD	GOOD	EXCELLENT	GOOD	EXCELLENT
Company Constraints	Cost	FREE	PAY	MIX	FREE	FREE	MIX	PAY
	Supported Platforms	Windows, Linux, macOS	Windows, Linux, macOS	Windows, Linux, macOS	Windows, Linux, macOS	Windows, Linux, macOS	Windows, Linux, macOS	Windows
	Complexity of License Management	EASY	ADEQUATE	EASY	EASY	EASY	MODERATE	ADEQUATE
	Easy to Install	YES	YES	YES	YES	YES	YES	YES
Railway-specific Criteria	CENELEC Certification	NO	PARTIAL	NO	NO	NO	NO	YES
	Integration into the CENELEC Process	MEDIUM	YES	MEDIUM	YES	YES	MEDIUM	YES

Figure 22 Summary of evaluation results (part 1)

Category	Name	FDR4	CPN Tools	CADP	mCRL2	SAL	TLA+	UMC
Development Functionalities	Specification/ Modeling	TEXTIN	GRA	TEXTIN	TEXT	TEXTIN	TEXT	TEXT
	Code Generation	NO	NO	YES	NO	NO	NO	NO
	Document / Report Generation	PARTIAL	PARTIAL	PARTIAL	PARTIAL	NO	NO	PARTIAL
	Requirements Traceability	NO	NO	NO	NO	NO	NO	NO
	Model refinement	YES	NO	NO	NO	NO	NO	NO
Verification Functionalities	Simulation	TEX	GRA	TEX	TEX	TEX	NO	TEX
	Supported problem size	LARGE	LIMITED	LARGE	MEDIUM	LARGE	MEDIUM	MEDIUM
	Formal Verification	RF	MC-B	MC-B	MC-B	MC-L, TP	MC-L, TP	MC-B
	Model-based Testing	NO	NO	YES	NO	YES	NO	NO
	Time related properties	YES	YES	NO	YES	YES	NO	NO
	Probability properties	NO	NO	NO	NO	NO	NO	NO
Language Expressiveness	Nondeterminism	INT,EXT	INT	INT,EXT	INT,EXT	INT,EXT	INT	INT
	Concurrency	ASYNCH	ASYNCH	ASYNCH	ASYNCH	ASYNCH, SYNCH	ASYNCH	ASYNCH, SYNCH
	Temporal Aspects	YES	YES	NO	NO	YES	NO	NO
	Probability	NO	NO	NO	NO	NO	NO	NO
	Modularity of the Language	HIGH	HIGH	HIGH	LOW	MEDIUM	MEDIUM	HIGH
	Supported Data Structures	COMPLEX	COMPLEX	COMPLEX	COMPLEX	COMPLEX	COMPLEX	COMPLEX
Tool Flexibility	Backward Compatibility	MODERATE	LIKELY	LIKELY	LIKELY	MODERATE	MODERATE	MODERATE
	Standard Input Format	YES	PARTIAL	YES	YES	YES	YES	YES
	Import/Export	MEDIUM	MEDIUM	HIGH	HIGH	MEDIUM	LOW	LOW
	Modularity of the Tool	LOW	LOW	LOW	LOW	LOW	LOW	LOW
	Team Support	NO	NO	NO	NO	NO	NO	NO
Maturity	Industrial Diffusion	LOW	MEDIUM	MEDIUM	LOW	LOW	MEDIUM	NO
	Stage of Development	YES	YES	YES	YES	YES	YES	NO
Usability	Customer Support	PARTIAL	PARTIAL	PARTIAL	PARTIAL	PARTIAL	PARTIAL	PARTIAL
	Graphical User Interface	LIMITED	PARTIAL	LIMITED	LIMITED	NO	LIMITED	PARTIAL
	Easy to Use	MEDIUM	MEDIUM	ADVANCED	ADVANCED	ADVANCED	ADVANCED	MEDIUM
	Quality of Documentation	EXCELLENT	GOOD	GOOD	GOOD	GOOD	GOOD	LIMITED
Company Constraints	Cost	MIX	FREE	MIX	FREE	FREE	FREE	FREE
	Supported Platforms	Windows, Linux, macOS	Windows	Windows, Linux, macOS	Windows, Linux, macOS	Windows, Linux, macOS	Windows, Linux, macOS	Windows, Linux, macOS
	Complexity of License Management	MODERATE	EASY	MODERATE	EASY	EASY	EASY	EASY
	Easy to Install	YES	YES	PARTIAL	YES	YES	YES	YES
Railway-specific Criteria	CENELEC Certification	NO	NO	NO	NO	NO	NO	NO
	Integration into the CENELEC Process	MEDIUM	MEDIUM	MEDIUM	LOW	LOW	LOW	LOW

Figure 23 Summary of evaluation results (part 2)

7 Ranking and Selecting Formal Methods

This section explains how the different information produced along the benchmarking task can be used to guide the selection of formal tools, based on the need of the user. The selection procedure is based on three main documents:

- **Tool Selection Support Matrix - Annex 1:** this document can be used to identify the formal tools used for a certain product and a certain phase, based on the interest of the user. This is useful to have a general idea on which types of formal languages and tools have been used in the development of a certain product and in a certain development phase;
- **Ranking Matrix - Annex 2:** this document can be used to provide a general ranking on the fourteen tools that have been carefully evaluated. The ranking is based on the user needs. Specifically, the user provides a set of scores to weight the different evaluation features considered. Based on these scores, the user is able to have a general ranking.
- **Tool Evaluation – Annex 3:** once promising tools are identified through the other annexes, this document presents a detailed analysis of the fourteen tools, and can be used to perform a more informed selection of the tool to use.

The following section will describe and explain, by means of examples, how to use the Tool Selection Support Matrix and the Ranking Matrix for the tasks illustrated above. The Tool Evaluation document is a MS Word document and does not need further explanation.

7.1 The Tool Selection Support Matrix

The Tool Selection Support Matrix collects all the papers inspected during the SLR, and consists of 114 rows, one for each paper, and a set of 11 columns, one for each feature of interest, such as “Phase” or “Category of Railway System”, plus other columns to identify the name of the paper, its Web-link and other information like number of citations and quality evaluation. The meaning of the different acronyms used in the columns is reported in Section 3.2.5.

The Tool Selection Support Matrix is a spreadsheet, and its users can select specific values of the considered features, by using the filtering function of the spreadsheet, depending on their needs.

For example, as illustrated in Figure 24 users may be interested in the development of ATP systems, and therefore they will filter the papers that include, in the column “Sub-category of Railway System”, the term “ATP”. Then, users may be interested in the Detailed Design (D) phase, and they will select from the column “Phases”, solely those papers that include the term “D”, as shown in Figure 26.

This filtering leads to three papers (see Figure 26), with corresponding tools (column Tool Name(s) in the spreadsheet, see an example on the left part of Figure 24). Users can inspect the linked papers, and understand in which way the different tools have been, and can be used.

Tool Name(s)	Task(s) (SM/SIM/ANA/FV /REF/MBT/CG/SA)	Phases (P/R/A/D/I/T/N/V/M)	Category of Railway System	Sub-category of Railway System	Type of Study	Industrial Evaluation (NO/LAB/IND/DEV)	Authorship (A/I/AI)
Simulink Design Verifier / Polyspace	SM / MBT / FV / CG / SA	R / A / D / I / T / V	SA	ATP			
Eclipse Modelling Framework (EMF) / Eclipse Modeling Framework Technology (EMFT) / Eclipse CDO / SEMCO model development tools (Semcmdt) / IBM's Rational Rhapsody Developer and Software Architecture / Arabion Editor	SM / REF	A	ERTMS / ETCS	Onboard ATP			
PHAVer	SM / FV	A	CTCS	ATP			
-	SM	A	NS	ATP	EX	NO	A
UPPAAL-TRON	SM / MBT	A / T	CBTC	ATP	EX	LAB	A
CPN Tools	SM / ANA / MBT	A / T	NS	Onboard ATP	ER	LAB	A

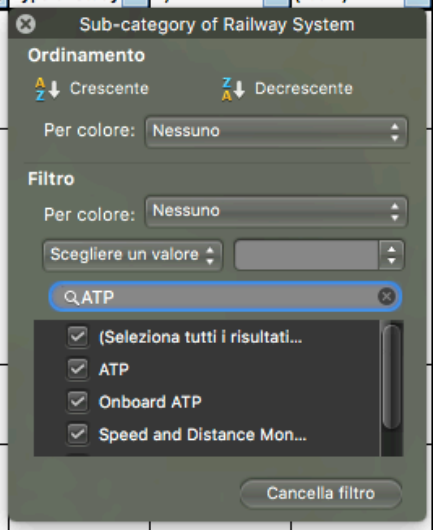


Figure 24 Selection of papers about ATP systems

Task(s) (SM/SIM/ANA/FV /REF/MBT/CG/SA)	Phases (P/R/A/D/I/T/N/V/M)	Category of Railway System	Sub-category of Railway System	Type of Study
SM / MBT / FV / CG / SA	R / A / D / I / T / V			
SM / REF / CG	A / D			
SM / FV / MBT / CG	A / D / T / V			

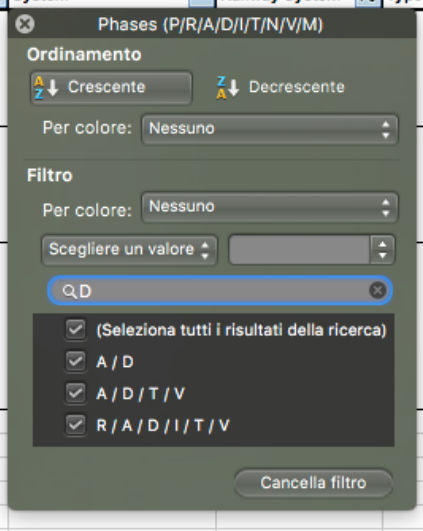


Figure 25 Selection of papers dealing with the Design (D) phase

ID	Paper Title	Source Library	Citations (Google scholar)	Quality Score	Formality (F/SF/SFF)
21	The Metrô Rio case study	ScienceDirect	12	13 Excellent paper reporting in detail on the actual application of formal methods and tools in the railway industry.	SFF
57	Using B as a High Level Programming Language in an Industrial Project: Roissy VAL	SpringerLink CS	132	11 Nice and extensive industrial experience with B: from modelling and theorem proving (type checking and refinement) to code generation	F
76	Lessons Learnt from the Adoption of Formal Model-Based Development	SpringerLink CS	6	9 Well-written report on the adoption of code generation technology in a medium-sized railway signalling company. Lacks information on the modelling and analyses.	SFF

Figure 26 Papers about ATP in the Detailed Design (D) phase

7.2 The Ranking Matrix

In the tables shown in Section 6.4 we have summarised the results of the experimentation and evaluation of the 14 selected tools with respect to a group of 34 evaluation features, divided into functional, language related, and quality aspects. The evaluation results have been summarised with a set of keywords that indicate, for each evaluation, the way in which the tool is positioned with respect to that aspect.

In order to pave the way towards a quantitative expression of the fitness of a tool for the desired purposes, a numerical value needs to be associated to each of the keywords appearing in the table.

Therefore we define a scale of values in the range 0 … 10 that would reflect our degree of appreciation for the way in which the tool supports a certain aspect. In particular, we suppose that a value of "0" indicates absolute non-satisfaction on how the tool deals with a certain aspect, that a value of "5" indicates a minimal level of appreciation, and a value of "10" as the maximum level of appreciation of the tool's features.

APPRECIATION DEGREES	
absolutely not satisfying	0
not satisfying	2,5
just acceptable	5
very nice to have	7,5
wonderful to have	10

After the appreciation degrees have been specified, it is possible to associate to each considered aspect and evaluation keyword its corresponding appreciation degree.

This has been done by extending the tables of Section 6.4 with 6 additional columns, in which for each aspect and keyword, a specific appreciation degree has been given.

Let us consider, for example, the first aspect of our verification related features, i.e., if and how the tool allows the user to run *simulations* of the system under design. In this case, the tools have been classified according to the keywords "NO" (meaning that the tool does not allow to execute system simulations), "TEX" (meaning that the tool allows to run and display simulations with a textual interaction process), "GRA" (meaning that the tool supports simulation of the system evolution using a graphical interface, e.g. by clicking

on buttons and observing the effects in graphical terms, like a sequence diagram drawing), and "MIX" (meaning a mixed approach in which, for example, the interactive evolution choices are selected through buttons, but the effects are presented in textual form). Our appreciation degree with respect to this aspect could be reflected by the values shown in Figure 27, expressing that we would consider extremely unsatisfying the absence of these features, and while appreciating its presence in any of its textual or graphical forms.

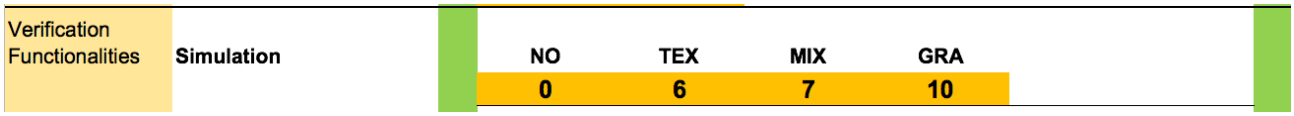


Figure 27 Appreciation degrees for the simulation feature

As a second example we could consider the way in which the modelling language would allow to describe non-deterministic aspects of the design. In this case the classification is based on the keywords "NO" (meaning that no forms of non-determinism are supported), "EXT" (meaning that non-determinism is supported in the form of interaction from the external environment, like external signals or input values), "INT" (meaning that non-deterministic internal choices are allowed, e.g. expressing alternative behaviours allowed by possible implementations). In this case, our appreciation degrees, as shown in Figure 28, could express that the absence of any form of nondeterminism would be extremely unsatisfying, while its presence in any form would be acceptable, with a greater appreciation in the case of support of the "internal" form of non-determinism and an even greater appreciation when both forms are supported.

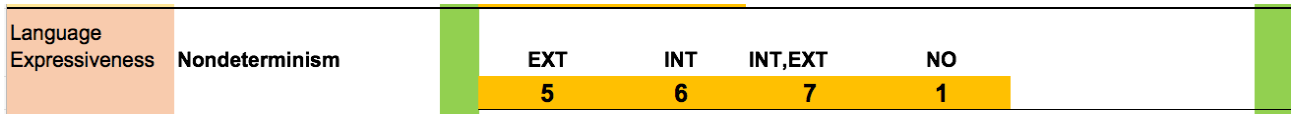


Figure 28 The appreciation degrees for the nondeterminism feature

As a third and last example we could consider the way in which tools allows the editing of the system models (Figure 29). In this case, the classification is based on the keywords "TEXIN" (meaning that the tool does not deal at all with the editing phase of the models, e.g. it is just a verification engine and the model editing is supposed to be done offline with a plain text editor), "TEXT" (meaning that the framework provides some kind of modelling language aware text editor to facilitate the editing), and "GRA" (meaning that the framework allows the editing of models using just boxes and arrows, or other custom graphical items, without an explicit textual representation of the design). In this case, our appreciation degrees might reflect that any framework choice is acceptable without problems, with some appreciation of the fact that non textual encoding are supported.

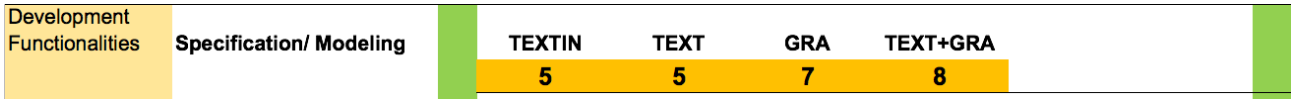


Figure 29 The appreciation degrees for the Specification / Modeling feature

From the above three cases we can already see that there can be a certain amount of arbitrariness in the mapping of keywords into the numeric satisfaction degree. For example, it is definitely possible that the enforcement of graphical editing of the models might be considered by someone an annoyance rather than a useful feature. This is the reason why in our final ranking matrix the satisfaction degree columns have been

filled with some reasonable initial values, but left fully editable by the user to allow their adjustments according to the user preferences.

Once all the keywords have been mapped into an appreciation degree, we are still not ready to build a global ranking of the selected tools. Indeed we have to consider that the 34 different aspects under which the tools have been compared are likely to have very different degrees of importance from the point of view of the user.

For this reason, in our Ranking Matrix an additional column has been added that allows to express, in a scale from 0 to 10, the importance that the user gives to the various aspects (Figure 30 shows the importance degrees in numerical form).

IMPORTANCE DEGREES	
completely irrelevant	0
not very important	2,5
medium importance	5
quite important	7,5
extreme importance	10

Figure 30 Importance degrees in numerical form

In Figure 31 is shown a possible example of assignment of importance degrees to the 34 tool features/aspects. In our final Ranking Matrix, however, the values associated with the importance of the tool features/aspects have been left non initialised (i.e. with value "0") because this mapping is extremely dependent from the user goals, environment and attitude.

For example, a user might be mainly interested in the use of formal tools for debugging of the early stages of the requirement-collection/system-design phases. In this case the presence of *Code Generation* features might be considered not very important, while a rich presence of *Verification* functionalities might be considered important (but maybe not necessarily the *time*-related/*probability*-related aspects).

For example in an academic research context the *Cost* might be considered an important aspect (greatly preferring open-source of free-licence tools), as well as the presence of a *Standard* (open) *Input Format* for the models, and the possibility to *Import/Export* models towards other tools/formats. Instead, all these aspects might not be considered crucial in an industrial environment, where greater importance is likely to be given to the presence of *Customer Support,* to the *Document / Report Generation*, and to all the tool *Maturity* related aspects.

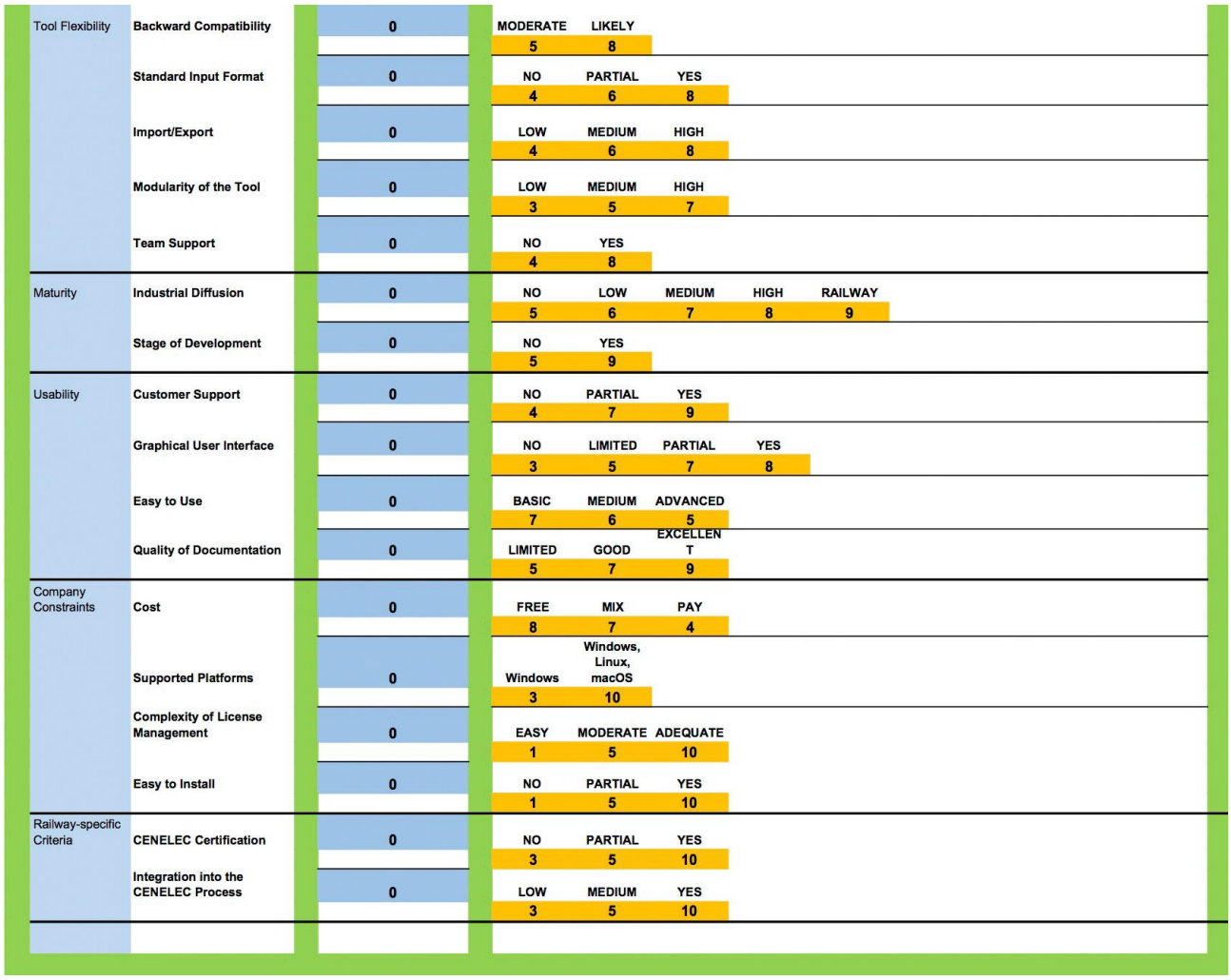
For this reason, the final structure of the Ranking Matrix will be like the one shown in Figure 32 and Figure 33, where all the feature importance degree have to be set by the user, and the appreciation degree for the specific features can be adjusted by the user, if necessary.

Category	Name	0 .. 10 IMPORTANCE
Development Functionalities	Specification/ Modeling	3
	Code Generation	1
	Document / Report Generation	6
	Requirements Traceability	1
	Model refinement	2
Verification Functionalities	Simulation	10
	Supported problem size	7
	Formal Verification	8
	Model-based Testing	2
	Time related properties	1
	Probability properties	2
Language Expressiveness	Nondeterminism	5
	Concurrency	8
	Temporal Aspects	1
	Probability	2
	Modularity of the Language	5
	Supported Data Structures	6
Tool Flexibility	Backward Compatibility	1
	Standard Input Format	2
	Import/Export	5
	Modularity of the Tool	4
	Team Support	1
Maturity	Industrial Diffusion	4
	Stage of Development	4
Usability	Customer Support	4
	Graphical User Interface	3
	Easy to Use	4
	Quality of Documentation	6
Company Constraints	Cost	6
	Supported Platforms	7
	Complexity of License Management	2
	Easy to Install	4
Railway-specific Criteria	CENELEC Certification	6
	Integration into the CENELEC Process	6

Figure 31 The column setting the desired feature importance

Category	Name	0 .. 10 IMPORTANCE	0 .. 10 APPRECIATION DEGREES								
Development Functionalities	Specification/ Modeling	0	TEXTIN	TEXT	GRA	TEXT+GRA					
			5	5	7	8					
	Code Generation	0	NO	PARTIAL	YES						
			3	5	7						
	Document / Report Generation	0	NO	PARTIAL	YES						
			5	6	7						
	Requirements Traceability	0	NO	YES							
			5	7							
	Model refinement	0	NO	YES							
			5	7							
Verification Functionalities	Simulation	0	NO	TEX	MIX	GRA					
			0	6	7	10					
	Supported problem size	0	LIMITED	MEDIUM	LARGE						
			4	6	8						
	Formal Verification	0	RF	MC-I	TP	MC-L	MC-B	TP, RF	MC-L, TP	MC-L,MC-B	MC-L,MC-B, TP, RF
			5	5	5	6	7	6	7	8	9
	Model-based Testing	0	NO	YES							
			4	7							
	Time related properties	0	NO	YES							
			4	7							
	Probability properties	0	NO	YES							
			4	7							
Language Expressiveness	Nondeterminism	0	EXT	INT	INT,EXT	NO					
			5	6	7	1					
	Concurrency	0	NO	SYNCH	ASYNCH	ASYNCH, SYNCH					
			4	4	6	8					
	Temporal Aspects	0	NO	YES							
			4	6							
	Probability	0	NO	YES							
			4	6							
	Modularity of the Language	0	LOW	MEDIUM	HIGH						
			4	6	8						
	Supported Data Structures	0	LIMITED	BASIC	COMPLEX						
			4	7	9						

Figure 32 All the custom evaluation parameters (part 1)



Tool Flexibility	Backward Compatibility	0	MODERATE	LIKELY				
			5	8				
	Standard Input Format	0	NO	PARTIAL	YES			
			4	6	8			
	Import/Export	0	LOW	MEDIUM	HIGH			
			4	6	8			
	Modularity of the Tool	0	LOW	MEDIUM	HIGH			
			3	5	7			
	Team Support	0	NO	YES				
			4	8				
Maturity	Industrial Diffusion	0	NO	LOW	MEDIUM	HIGH	RAILWAY	
			5	6	7	8	9	
	Stage of Development	0	NO	YES				
			5	9				
Usability	Customer Support	0	NO	PARTIAL	YES			
			4	7	9			
	Graphical User Interface	0	NO	LIMITED	PARTIAL	YES		
			3	5	7	8		
	Easy to Use	0	BASIC	MEDIUM	ADVANCED			
			7	6	5			
	Quality of Documentation	0	LIMITED	GOOD	EXCELLENT			
			5	7	9			
Company Constraints	Cost	0	FREE	MIX	PAY			
			8	7	4			
	Supported Platforms	0	Windows	Windows, Linux, macOS				
			3	10				
	Complexity of License Management	0	EASY	MODERATE	ADEQUATE			
			1	5	10			
	Easy to Install	0	NO	PARTIAL	YES			
			1	5	10			
Railway-specific Criteria	CENELEC Certification	0	NO	PARTIAL	YES			
			3	5	10			
	Integration into the CENELEC Process	0	LOW	MEDIUM	YES			
			3	5	10			

Figure 33 All the custom evaluation parameters (part 2)

Once all features importance degrees and feature support appreciation degrees have been defined we have the possibility to compute a possible evaluation score for each tool/framework.

This global score of a tool can be computed as the sum, for all 34 features, of the product of the feature importance times the specific appreciation degree for that specific tool and associated feature. This final score can then be normalised in a scale 0...100 (as shown in Figure 34) by dividing the result of the sum for the maximum possible achievable score[14] and by multiplying the result by 100.

The customizable spreadsheet containing all the data and ranking formulas are provided in Annex 3.

[14] Sum of all the importance degrees * 10

FINAL RANKING SCALE	
absolutely not satisfying	0
not satisfying	25
just acceptable	50
very nice to have	75
wonderful to have	100

Figure 34 Ranges of the final scores

7.3 Summary and Discussion

In this section, we presented the approach to be followed for the selection of the most adequate formal or semi-formal methods to be used for the development of a certain railway product. The main focus is on the architecture and design phase of the CENELEC process, since most of the tools address this phase. However, information is available to identify tools also for the other phases of the process. We do not provide a final ranking of the different tools evaluated, since the ranking highly depend on the needs of the company, and it would be misleading to create an explicitly ranked list of tools. Instead, we provide a Ranking Matrix (Annex 2) that can be used to support the selection, and that can be tuned by the company based on its specific needs. The company can select appropriate weights for the different evaluation criteria, and the ranking formulas in the matrix will generate a ranking based on (a) the selected weights and (b) the evaluation values assigned to the evaluation criteria as described in Section 6. It should be noticed that the Ranking Matrix alone is not sufficient to perform an informed selection, and its users are encouraged to consider the information provided by Tool Selection Support Matrix (Annex 1) and by the Tool Evaluation document (Annex 3).

8 Conclusions

The current deliverable is the report of T4.1 - Benchmarking and T4.2 - Ranking of Work-package 4 of the ASTRail project. The document presents a set of activities aimed at supporting the identification of the most suitable formal and semi-formal methods to be used for railway systems development. Specifically, a systematic literature review was conducted to categorise 114 scientific publications on formal methods and railways according to features such as the type of system and the phase of the development process addressed by the experience considered in the publication. The literature review was complemented with a project review and a survey with practitioners, to identify the most mature formal and semi-formal methods and tools to be used in a railway context. This analysis has shown a dominance of the UML modelling language for high-level representation of system models, and a large variety of formal tools used, with a dominance of the tools associated to the B family (ProB and Atelier B), followed by several other tools, including Simulink, NuSMV, Prover, SCADE, IBM Rational Software Architect, Polyspace, S3, SPIN, CPN Tools, etc. The project review and the survey with practitioners confirmed this scattered landscape. According to the survey with practitioners, one of the most relevant features that a tool should support was considered formal verification, and, therefore, a set of tools supporting both modelling and formal verification was considered for accurate experimentation and evaluation. A set of 14 tools, considered as the most promising, was carefully reviewed by means of a systematic evaluation based on a set of 34 evaluation features.

The final product of these activities is a set of informative documents to support the ranking and selection of formal and semi-formal methods for railways, based on (a) the information retrieved from the literature, summarised in a Tool Selection Support Matrix (Annex 1), (b) the information available from the tools' evaluation (Annex 3), and (c) the Ranking Matrix (Annex 2), which allows users to weight the different evaluation criteria, and come to a fine-grained selection of the most appropriate formal methods and tools, suitable to their needs.

Acronyms

Acronym	Explanation
SLR	Systematic Literature Review
ATP	Automatic Train Protection
ATO	Automatic Train Operation
RBC	Radio Block Centre

List of figures

List of tables

References

[Scu97] Scupin, R. (1997). The KJ method: A technique for analyzing data derived from Japanese ethnology. *Human organization*, *56*(2), 233-237.

[BP06] Budgen, D., & Brereton, P. (2006). Performing systematic literature reviews in software engineering. In *Proceedings of the 28th international conference on Software engineering* (pp. 1051-1052). ACM.

[KBB09] Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology*, *51*(1), 7-15.

[DD08] Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and software technology*, *50*(9-10), 833-859.

[CB11] Chen, L., & Babar, M. A. (2011). A systematic review of evaluation of variability management approaches in software product lines. *Information and Software Technology*, *53*(4), 344-362.

[CEN11] CENELEC, EN 50128 (2011). *Railway applications-Communication, Signaling and Processing Systems-Software for Railway Control and Protection Systems*.

[ISO25010] ISO/IEC, 25010:2011 (2011) *Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models*.

[CLA96] Clarke, E. M., & Wing, J. M. (1996). Formal methods: State of the art and future directions. *ACM Computing Surveys (CSUR)*, *28*(4), 626-643.

[BKM08] Bangor, A., Kortum, P. T., & Miller, J. T. (2008). An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, *24*(6), 574-594.

[BOC09] Boca, P., Bowen, J. P., & Siddiqi, J. (Eds.). (2009). *Formal methods: State of the art and new directions*. Springer Science & Business Media.

[CRA95] Craigen, D., Gerhart, S., & Ralston, T. (1995). *Industrial Applications of Formal Methods to Model, Design and Analyze Computer Systems - An International Survey*. Noyes Data Corporation.

[MON12] Monin, J. F. (2012). *Understanding formal methods*. Springer Science & Business Media.

[WOD12] Woodcock, J., Larsen, P.G., Bicarregui, J., & Fitzgerald, J. (2009). Formal methods: Practice and experience. *ACM Computing Surveys*, 41(4): 1-36.

[BBFM99] Behm, P., Benoit, P., Faivre, A., & Meynadier, J. M. (1999). METEOR: A successful application of B in a large project. In *International Symposium on Formal Methods* (pp. 369-387). Springer, Berlin, Heidelberg.

[TWC01] Tretmans, J., Wijbrans, K., Chaudron, M.R.W. (2001). Software Engineering with Formal Methods: The Development of a Storm Surge Barrier Control System Revisiting Seven Myths of Formal Methods. *Formal Methods in System Design*, 19(2): 195-215.

[FAN13] Fantechi, A. Twenty-five years of formal methods and railways: what next?. In *International Conference on Software Engineering and Formal Methods* (pp. 167-183). Springer, Cham.

[MBK14] Marshall, C., Brereton, P., & Kitchenham, B. (2014, May). Tools to support systematic reviews in software engineering: A feature analysis. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering* (p. 13). ACM.

[KLL97] Kitchenham, B., Linkman, S., & Law, D. (1997). DESMET: a methodology for evaluating software engineering methods and tools. *Computing & Control Engineering Journal*, *8*(3), 120-126.

[AFPM11] Bacelar Almeida J., Frade M. J., Sousa Pinto J. & Melo de Sousa S (2011). An Overview of Formal Methods Tools and Techniques in Rigorous Software Development - An Introduction to Program Verification. Undergraduate Topics in Computer Science, Springer, 15--44.

[BGK18] ter Beek M.H., Gnesi S. & Knapp A. (2018). Formal methods for transport systems. International Journal on Software Tools for Technology Transfer, Springer, 237--241.

[OR17] O'Regan G. (2017). Concise Guide to Formal Methods - Theory, Fundamentals and Industry Applications, Undergraduate Topics in Computer Science, Springer.

Tool Evaluation

Annex ID	D4.1 Annex 3
Title	Tool Evaluation
Work Package	WP4
Dissemination Level	PUBLIC
Version	1.0
Date	2018-05-16
Status	Final
Lead Editor	ISTI
Main Contributors	ISTI

Published by the ASTRAIL Consortium

Document History

Version	Date	Author(s)	Description
0.1	2018-03-21	ISTI	First Draft
1.0	2018-05-16	ISTI	Final

Table of Contents

1 Tool Evaluation

All the tools/frameworks considered deal with the construction, refinement and verification of specification models. i.e. these tools are mainly applicable in the Detailed-Design phase.

However, their applicability can sometimes interact with other SW Development phases. In particular, it is possible that they directly lead to the executable program code generation.

Or it is possible that they trace the user requirements though all the design (of program code) steps. Finally, is it possible that these tools provide some support for the testing phases of the software development (see **Development Functionalities**).

The tools/frameworks may greatly differ in the way in which the models can be constructed. Their specification language can be oriented to the definition of single state machine, or to the definition of a set of interacting components. The system can be modelled as a closed nondeterministic system, or as an open event-triggered, system. The specification style can be logical, functional, imperative, algebraic. The data types manipulated by the model can be simple elementary types, structured types, generic abstract data types, or even high order data.

The framework might allow the flat definition of a specification model, or a whole hierarchy of refinements that specify the system at different levels of abstraction. A model definition might be defined in a compositional/modular way or not. The specification language might have constructs to describe time dependent aspects, or even probabilistic aspects. Some of the above aspects (e.g. specification style) cannot be directly associated to an absolute quality measure, however their knowledge might be useful to understand the potential impact on the current software development process (see **Language Expressiveness**).

The tools frameworks may be based on different underlying formal techniques. They can be based on model checking techniques (BDD based, SAT/SMT based, explicit, on the fly), or on theorem proving techniques. The tool might allow the interactive (or random) system simulation, and allow support forms of model based testing.

It is not uncommon that the formal frameworks allow the verify the equivalence or the specification-implementation relation between two models.

The system properties that the tools allow to analyse can sometimes be expressed as invariants over the possible states of the system, or as temporal properties over the sequences of states that can be traversed during a system execution, or as temporal properties over system evolutions seen as a tree of computation steps.

In some cases, it is even possible to express the system properties with the full power of mathematical and first order logics. Some of the above aspects can apparently be only informative (e.g. the underlying formal technique on which the tools are based), but this information may actually have an implicit impact in terms of performance (e.g. bdd based model checking vs explicit model checking) or functionality (capability to verify infinite states systems) (see **Verification Functionalities**).

The industrial readiness of the tool/framework, however, also depends from other characteristics non directly associated to model constructions and verification issues.

The development of critical system would ideally require the use of fully qualified toolsets. This situation is rarely encountered in practice, therefore it becomes important for the toolset to show its correctness through the maturity of the process by which it has been developed, and by the log of industrial usages through which is has successfully undergone. The use of advanced academic prototypes is surely not forbidden, but the knowledge of the actual maturity status of the framework show be clearly acknowledged (see **Degree of Maturity**).

Since critical industrial project, and railways are no exception, usually have a long time of exercise, it is very important that the continuous flow of new versions of the tool does not create problems in the maintenance of the development environment. It is therefore important that the tool/framework is able to deal with legacy version of the models even in the case of tool evolutions.

Not all the desired analysis features might be made available by a single tool/framework, it becomes therefore important the possibility of the framework to export its models in an open format potentially usable (or convertible) by other tools. Similarly, it is a plus if the internal models used by the framework are based on a public format, so that it is possible to import (or translate) models from other frameworks. (see **Flexibility**)

It is perfectly acceptable in an industrial context the presence of fee-based licenses, but a flexible management of them, maybe tuned on the specific set of components actually needed by the client is definitively a plus. Also the overall cost of the framework is certainly an important evaluation parameter.

It would be highly preferable that the introduction of tool/framework had a smooth impact on the existing company environment, e.g. not requiring hardware platforms not present in the company, or requiring an installation procedure that interferes (e.g. in terms of libraries) with the company development environment. (see **Company Constraints**).

In an industrial context, being industry-ready means also able to provide adequate support to client, both in terms of maintenance of the tool, and in terms of available tool documentation and training. Being the verification tool/framework based on potentially complex mathematical/logical foundations, the degree of knowledge/competence required by the users to use the tools must be adequate to the company profile. The presence of a graphical user interface of good quality is often an aspect that facilitates the introduction of the tools in the industrial context (see **Usability**).

Finally, since our survey is focused on the railway sector, the tool/framework support of specific sectorial standards specific sectorial standard should be taken into account (see **Railway-Specific Criteria**).

2 The Evaluation Sheet Template

_____Information Part_____

Tool/Framework Name:

Description:

Web Sites:

Documentation:

Reports on Industrial Uses of the Tool (in Railways):

_____ Evaluation Part _____

Development Functionalities

How the framework supports the construction and refinement of specification models, their translation onto executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Textual, Graphical, TextualInport**

Textual: Models are edited with the tool in plain textual form.

Graphical: Models are edited with the tool through a graphical interface.

Textualinport: The tool just operates on textual data provided by other tools.

Code Generation: **Yes, No**

Yes: The tool supports the automatic generation of program code from the models.

Document / Report Generation: **Yes, No, Partial**

Yes: The tool supports automated generation of readable reports and documents, which describe the artifacts produced with the tool, or the activities carried out with the tool.

No: Feature not mentioned.

Partial: The tool allows to generate diagrams or partial reports that can be in principle included in official documentation.

Requirements Traceability: **Yes, No**

Yes: The tool supports traceability of requirements to the artifacts produced with the tool.

Model Refinement: **Yes, No**

Yes: The framework supports the definition of a hierarchy of models at different levels of abstraction.

Verification Functionalities

The approaches used by the framework to verify the models and the kind of properties that the frameworks allows to verify.

Simulation: **No, Graphical, Textual, Textual+Graphical**

Graphical (GRA): the visualisation of the simulation is purely in the form of visual diagrams

Textual (TEX): the visualisation of the simulation is purely in textual format

Textual-Graphical (MIX): the visualisation of the simulation is mainly textual but it is aided by visual diagrams

No (NO): the tool does not support simulation of the possible system evolutions

Formal Verification: **Refinement Checking, Theorem Proving, Model Checking (Invariants, Linear, Branching)**

Refinement Checking(RF): Allows to verify equivalence / refinement relations among models

Model Checking Invariants (MC-I): Allows to verify invariant state properties along the possible evolution graph of the system.

Model Checking Linear (MC-L): Allows to verify linear time properties along the possible evolution graph of the system (this includes model checking of invariants as a subcase).

Model Checking Branching (MC-B): Allows to verify branching time properties along the possible evolution graph of the system (this includes model checking of invariants as a subcase).

TheoremProving(TP): Allows to automatically verify logical properties of the model or to mechanically verify theorem proofs over the model.

Supported problem size: **Limited, Medium, Large**

Limited: theorem proving, model checking of system with less than a few million states

Medium: model checking model checking of system with less than a few hundred million states

Large: model checking infinite state model checking, systems with many hundred million (billions) of states.

Model Based Testing: **Yes, No**

Yes: The framework supports the automatic generation of test

No: The framework allows the construction of testing oracles

Time Related Aspects: **Yes, No**

Yes: the tool allows the verification of time related properties

No: the tool does not support the verification of time related properties

Probability Aspects: **Yes, No**

Yes: the tool allows the verification of probability related aspects

No: the tool does not support the verification of probability related aspects

Property Specification Language: **name (informative)**

Language Expressiveness

The characteristics of the models that can be generated within the framework.

Name of Language: **name**

(just informative aspect)

Nondeterminism: **Internal, External, No**

Internal: the model allows nondeterministic system evolutions

External: the model allows nondeterminism associated to external inputs or trigger-events

No: the model is fully deterministic

Concurrency: **Async, Sync, No**

Asynch: The model can be constituted by a set of asynchronously interacting elements

Sync: The model can be constituted by a set of synchronous elements

No: The model is constituted by just one element

Temporal aspects: **Yes, No**

Yes: The language of the tool supports the notion of time

Notice: A tools may support time notations (for code generation or simulation oriented) in the design, without being able to prove any time-related property, or not support time related notations for the modelling, while still allowing the verification of time-like properties expressed e.g. in terms of evolution steps.

Stochastic aspects: **Yes, No**

Yes: The language of the tool supports the notion of probability

Modularity aspects: **High, Medium, Low**

High: The tool allows the user to model in a hierarchical way, and the partitioning of the model Into modules

Medium: The tool allows the partitioning of the model into modules but does not allow the user to model in a hierarchical way.

Low: The tool allows the partitioning of models into modules, but the modules have no way to interact, neither by messages nor by shared memory.

Supported Data Structures: **Minimal, Basic, Complex**

Minimal: The language supports just booleans and integer values.

Basic: The language supports various kind of numeric types, but no composite expressions

Complex: The language supports complex expressions like sequences, sets, array values.

Model kind: **Imperative, Functional, Algebraic, Logical, Graphic**

(just informative aspect)

Imperative: the model is described by a textual imperative language

Functional: the model is described by a textual functional language

Algebraic: the model is described by a textual process algebra

Logical: the model is described by a textual logical language

Graphic: the model is described by a graphical notation

Flexibility

Backword Compatibility: **Yes, Likely, Moderate, Uncertain**

Yes: The vendor guarantees that legacy versions of the models can be used in the current version of the tool or the future availability of legacy versions of the tool.

Likely: The tool is open source, or the input language is stable and standard de facto or there is evidence of interest in preserving backward compatibility.

Moderate: The tool is not open source, and the provider does not show evidence regarding the backward compatibility, even is the language is rather stable and standard de facto.

Uncertain: Sources not available, input format not necessarily stable, no information available from vendor.

Standard Input Format: **Yes, No, Partial**

Yes: The input language is not proprietary, open and public.

Partial: The structure of the model specifications is easily accessible, but not publicly documented.

No: The internal structure of the model specification is hidden.

Import/Export to other tools: **Medium, Low, High**

High: The tool provides several inport/export functionalities

Medium: The tool has a standard format used by other tools, or exports w.r.t. to other formats.

(Knows cases of interactions with are tools are mentioned.)

Low: Tool not oriented towards eport/export functionalities

Modularity of the tool: **High, Medium, Low**

High: The tool includes is composed of many packages that can be loaded to address different phases of the development process

Medium The tool can address different phases of the development process in a monolithic way.

Low The tool is used in only a single development phase.

Team Support: **Yes, No**

Yes: The tool supports collaborative team development

Maturity

Industrial Diffusion: **Railway, High, Medium, Low**

Railway: Tool widely used in railway and specifically oriented to railway systems.

High: the tool is oriented towards industrial usage. It is widely used in industry, there is partial evidence of its usage in railways, but the tool is not specifically oriented to railway systems.

Medium: The the tool is not specifically oriented towards industrial usage. The tool is widely used in industry, there is partial evidence of its usage in railways, but the tool is not specifically oriented to railway systems.

Low: The tool is partially used in industry, but without evidence of its usage in railways.

No: there are no known cases of use of the tool in industry

Stage of Development: **Yes, No, Partial**

Yes: The tool is a stable product with a long history of versions

Partial: The tool is a recent tool but with a solid infrastructure

No: The tool is at the level of a prototype

Usability

Availability of Customer Support: **Yes, No, Limited, Partial**

Yes: Reliable customer support can be acquired for maintenance and training

Partial: Free support is available for maintenance and training (e.g. mail for bug notifications and public forums for discussions)

No: communications channels are established among producers and users

Graphical User Interface: **Yes, Partial, Limited, No**

Yes: the tool has a well defined and powerful graphical user interface

Partial: a user friendly GUI exists, but does not cover all the tool functionalities in a graphical form

Limited: a GUI exists, but not particularly effective in the design and usability

No: The tool is a command line tool.

Easy of Use: **Basic, Medium, Advanced**

Which mathematical background is needed for an effective use of the tool

Basic: the tool does not require particular logical/mathematical skills

Medium: the tool requires some degree mathematical/logical skills not difficult to find among domain experts

Advanced: the effective use of the requires advanced logical/mathematical skills.

Quality of Documentation: **Excellent, Good, Limited**

Excellent: The documentation is extensive, updated and clear, and includes examples that can be used by domain experts, and it is accessible and navigable in an easy way

Good: The documentation is complete, but offline and requires some effort to be navigated

Limited: The documentation is not sufficient, or easily accessible, to effectively use the tool, but that activity can still be finalised with additional effort

Company Constraints

Cost: **Pay, Free, Mix**

Pay: Available only under payment

Mix: Free under limited conditions (academic) and moderate cost for industrial uses.

Free: Free for all industrial or academic uses

Supported Platforms: **names of platforms**

names of platforms: the names of the supported platforms (macOS, Windows, Linux, Solaris,...)

Easy and Flexible License Management: **Easy, Moderate, Adequate, Complex**

Easy the tool is free for commercial use, and no license management system is required

Moderate the tool has a free version and a commercial one. While trying the tool with a free license, we did not encounter any licensing problem. Limited information is provided concerning the licensing system for

commercial licenses. (The underlying assumption is that the license management for commercial licenses will be sufficiently easy, as experimented for free licenses).

Adequate the tool is only available upon payment. When trying the tool with the academic license we encountered a limited overhead in dealing with licenses. The licensing information provided in the website of the tool is clear and accurate.

Complex the tool is only available upon payment. The tool could not be tried, and the information provided by the website of the tool concerning license management is limited.

Easy to Install: **Yes, No, Partial**

Yes: The tool is mostly self contained, does not require external libraries, and can be easily installed.

Partial: The tool installation depends on external components, and the installation process is not smooth

No: The installation process can interfere with the customer development environment.

Railway Specific Criteria

CENELEC certification: **Yes, No, Partial**

Yes: The tool is certified according to the CENELEC norm

Partial: The tool includes a CENELEC certification kit, or if the tool is certified according to other safety-related norms (e.g., DO128C)

No: otherwise

Integration into the CENELEC process: **Yes, Medium, Low**

Yes: the tool or language is mentioned in the text of the CENELEC norm, in the literature and in the tool documentation we found evidence of the usage of the tool for the development of railway products developed according to the CENELEC norms.

Medium: in the literature and in the tool documentation we found evidence of the usage of the tool in railways, but we did not find any evidence of CENELEC products developed with the support of the tool.

Low: in the literature and in the tool documentation we did not find any evidence of usage of the tool in railways.

3 SPIN

_____Information Part_____

Tool/Framework Name: SPIN

Description:

SPIN8 (Simple Promela Interpreter) is an advanced and very efficient tool specifically targeted for the verification of multi-threaded software. The tool was developed at Bell Labs in the Unix group of the Computing Sciences Research Center, starting in 1980. In April 2002 the tool was awarded the ACM System Software Award. The language supported for the system specification is called Promela (PROcess MEta LAnguage).

See also: https://en.wikipedia.org/wiki/SPIN_model_checker.

Web Sites:

http://spinroot.com

Documentation:

http://www.spinroot.com/spin/whatispin.html
http://spinroot.com/spin/Man/index.html
http://www.spinroot.com/spin/Man/GettingStarted.html
http://www.spinroot.com/spin/Man/promela.html
Book: Principles of the Spin Model Checker
Book: The SPIN Model Checker: Primer and Reference Manual
Many books, tutorials, slides, available online.

Reports on Industrial Uses of the Tool (in Railways):

"A Formal Specification and Validation of a Critical System in Presence of Byzantine Errors"
https://link.springer.com/chapter/10.1007/3-540-46419-0_36

"Towards Model-Driven V&V assessment of railway control systems"
https://link.springer.com/article/10.1007/s10009-014-0320-7

_____ Evaluation Part _____

Development Functionalities

How the framework supports the construction and refinement of specification models, their translation onto executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

SPIN is mainly a command line oriented model checker. It is not an integrated design/verification framework. The jSpin/iSpin GUI are provided by third parties, with little capabilites. The Promela specification language is the source/target of many (unsupported) translators provided by third parties.

Specification/Modeling: **Textual**

The tool "spin" is just a verification/analysis tool working on textual files.

The "jspin.jar" and "ispin.tcl" GUI allow also the editing of models.

Code Generation: **No**

Document / Report Generation: **Partial**

The ispin/jspin GUI allow to visualize execution diagrams generated interactively or through the guidance if a counter-example.

Requirements Tracebility: **No**

Model Refinement: **No**

Verification Functionalities

The approaches used by the framework to verify the models and the kind of properties that the frameworks allows to verify.

Simulation: **Textual**

Formal Verification: **Model Checking (Invarients, Linear)**

Supported problem size: **Large**

Model Based Testing: **No**

TimeRelatedAspects: **No**

ProbabilityAspects: **No**

Property Specification Language: **LTL**

(just informative)

Notes:

Verification allows to specify fairness constrains and assertions. LTL is state based. Built-in deadlock analysys.

Language Expressiveness

The characteristics of the models that can be generated within the framework.

Name of Language: **Promela**

Nondeterminism: **Internal**

The choice operator allows nondetermistic selection of transition rules.

A system is seen as a collection of processes. Processes interact through synchronous Message Passing towards buffered channel. The behavior of a process can be nondeterministic. The choice of which process is selected for progress is nondeterministic. Processes can share memory.

Concurrency: **Async**

A system is composed by a set of processes scheduled by interleaving.

Temporal aspects: **No**

Stochastic aspects: **No**

Modularity aspects: **High**

A system is composed by a dynamic set of hierarchical processes.

Supported Data Structures: **Basic**

Expressions can only be of elementary types, statically sized arrays variables are allowed.

Model kind: **Imperative**

Flexibility

Backword Compatibility: **Lilkely**

The modelling language is a de facto standard and very stable. The tools is also open source.

Standard Input Format: **Yes**

Import/Export to other tools: **Medium**

Actually there is no need of built-in import/export functionalities, since the design language is open.

In the literature are mentioned many cases of convertion from/to promela models.

Modularity of the tool: **No**

Team Support: **No**

Maturity

Industrial Diffusion: **Medium**

Many cases of industrial uses.

Stage of Development: **Yes**

Usability

Availability of Customer Support: **Partial**

http://spinroot.com/fluxbb/ SPIN Public Discussion Forum

Graphical User Interface: **Limited**

Two GUI exist, jSpin and iSpin. But design functionalities are limited.

Easy of Use: **Medium**

Properties are encoded as LTL formulae, possibly with fairness constrains.

Quality of Documentation: **Good**

Company Constraints

Cost: **Free**

Free software BSD-3Clause license

see http://www.spinroot.com/spin/spin_license.html

Supported Platfoms: **macOS,Windows,Linux,Solaris,...**

Easyand Flexible License Management: **Easy**

Easy to Install: **Partial**

Requires Developers Tools with Command Line extensions (e.g. gcc compilation system).

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Medium**

The tools appear to be used in the railway field but without explicit references to CENELEC related activitites.

Notes

4 Simulink

_____Information Part_____

Tool/Framework Name: Simulink

Description:

Simulink, developed by MathWorks, is a graphical programming environment for modeling, simulating and analyzing multidomain dynamical systems. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries. It offers tight integration with the rest of the MATLAB environment and can either drive MATLAB or be scripted from it. Simulink is widely used in automatic control and digital signal processing for multidomain simulation and Model-Based Design

Web Sites:

https://it.mathworks.com/products/simulink.html

Documentation:

https://it.mathworks.com/help/index.html
https://it.mathworks.com/help/simulink/getting-started-with-simulink.html

Webinar: https://it.mathworks.com/videos/
/model-based-approach-for-ertms-railway-wayside-system-specification-validation-and-proof-90417.html

Reports on Industrial Uses of the Tool (in Railways):

"A Story About Formal Methods Adoption by a Railway Signaling Manufacturer"
https://link.springer.com/chapter/10.1007/11813040_13

"Contract Modeling and Verification with FormalSpecs Verifier Tool-Suite - Application to Ansaldo STS Rapid Transit Metro System Use Case"
https://link.springer.com/chapter/10.1007/978-3-319-24249-1_16

"The Metrô Rio case study"
https://www.sciencedirect.com/science/article/pii/S0167642312000676

_____ Evaluation Part _____

Development Functionalities

How the framework supports the construction and refinement of specification models, their translation onto executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Graphical**

Code Generation: **Yes**

Document / Report Generation: **Yes**

https://www.mathworks.com/products/SL_reportgenerator.html

Requirements Tracebiity: **Yes**

https://it.mathworks.com/discovery/requirements-traceability.html

Model Refinement: **Yes**

https://it.mathworks.com/discovery/model-based-testing.html

Verification Functionalities

The approaches used by the framework to verify the models and the kind of properties that the frameworks allows to verify.

Simulation: **Graphical**

Formal Verification: **Model Checking (Invariants)**

The model must be extended with "observer" components to highlight hte properties of interest (in the form of invariants).
The "Design Verifier" fuctionality calls a SAT based Bounded model checker.
https://it.mathworks.com/discovery/formal-verification.html
https://it.mathworks.com/products/sldesignverifier.html

Supported problem size: **Limited**

Model Based Testing: **TestGeneration**

Through Simulink Design Verifier it is possible to generate automatically tests, with a full coverage (https://it.mathworks.com/discovery/model-based-testing.html)

TimeRelatedAspects: **Yes**

ProbabilityAspects: **No**

Property Specification Language:

Language Expressiveness

The characteristics of the models that can be generated within the framework.

Name of Language: **Simulink**

Nondeterminism: **External**

Input signals and values may trigger alternative behaviors. Single transitions rules are deterministic.

Concurrency: **No**

Even if the model can be decomposed into a set of interacting components, the overall system semantics is that of a single sequential system.

Temporal aspects: **Yes**

Stochastic aspects: **No**

Modularity aspects: **High**

Supported Data Structures: **Complex**

Typical programming language types are supported (float, pointers)

Model kind: **Graphic**

Flexibility

Backword Compatibility: **Likely**

Old versions of the tool remain available and new versions of the model can be downgraded

(see https://it.mathworks.com/help/simulink/ug/saving-a-model.html)

Standard Input Format: **No**

Import/Export to other tools: **Low**

Its is theoretically possible to export the models into custom export formats.
(E.g. SCADE provides functionalities to import Simulink Models)

Modularity of the tool: **Yes**

Team Support: **No**

Maturity

Industrial Diffusion: **High**

Claimed to be (from "An Operational Semantics for Stateflow" by Gregoire Hamon and John Rush) one of the most widely used environments of this kind is the Matlab suite from Mathworks which, with more than 500,000 licensees, is widespread throughout aerospace, automotive, and several other industries, and ubiquitous in engineering education.

Stage of Development: **Yes**

Usability

Availability of Customer Support: **Yes**

Graphical User Interface:

Easy of Use: **Basic**

The tool is very rich and complex. Mastering it requires a deep training.

System properties are specified by graphically combining predefined operators.

No deep mathematical knowledges are needed.

Quality of Documentation: **Excellent**

The documentation for MatLab is very good, several topics are covered (see for an overall index: https://it.mathworks.com/help/index.html) and there is an active community, also because of the widespread usage of the tool (see Industrial Usage). Parts of the online documentation requires a client account.

However, it is mainly used by engineers and it is difficult sometimes to find answer to more theoretical questions, as for example the possibility of expressing and verifying temporal logic formulae within this framework.

Company Constraints

Cost: **Pay**

Supported Platfoms: **macOS, Windows, Linux**

Easy and Flexible License Management: **Adequate**

Student, academic, and commercial licenses, individual or by group, based on the set of needed functionalities are available. Detailed information is available

(see https://it.mathworks.com/pricing-licensing.html)

Easy to Install: **Yes**

Railway Specific Criteria

CENELEC certification: **Partial**

There are available kits for certifying software that has been created using this framework, in particular a DO Qualification Kit (for DO-178) and IEC Certification Kit (for ISO 26262 and IEC 61508) (both for code generation aspects) are available.

Integration into the CENELEC process: **Yes**

5 NuSMV/nuXmv

_____Information Part_____

Tool/Framework Name: NuSMV (nuXmv)

Description:

NuSMV is a reimplementation and extension of SMV symbolic model checker, the first model checking tool based on Binary Decision Diagrams (BDDs).[1] The tool has been designed as an open architecture for model checking. It is aimed at reliable verification of industrially sized designs, for use as a backend for other verification tools and as a research tool for formal verification techniques.

NuSMV has been developed as a joint project between ITC-IRST (Istituto Trentino di Cultura in Trento, Italy), Carnegie Mellon University, the University of Genoa and the University of Trento. Since version 2, it combines BDD-based model checking with SAT-based model checking.

Its last evolution, called nuXmv, allows the verifications also of infinite-state systems.

It is maintained by Fondazione Bruno Kessler, the successor organization of ITC-IRST.

Web Sites:

http://nusmv.fbk.eu/
https://nuxmv.fbk.eu/

Documentation:

http://nusmv.fbk.eu/NuSMV/userman/index-v2.html
http://nusmv.fbk.eu/NuSMV/tutorial/index.html
http://nusmv.fbk.eu/NuSMV/papers.html
https://es.fbk.eu/tools/nuxmv/downloads/nuxmv-user-manual.pdf

Reports on Industrial Uses of the Tool (in Railways):

http://es.fbk.eu/projects

"A formal systems engineering approach in practice: an experience report"
https://dl.acm.org/citation.cfm?id=2593850.2593856

"Formalization and validation of a subset of the European Train Control System"
https://dl.acm.org/citation.cfm?id=1810312

"Validation of requirements for hybrid systems: A formal approach"
https://dl.acm.org/citation.cfm?id=2377659

"A Story About Formal Methods Adoption by a Railway Signaling Manufacturer"
https://link.springer.com/chapter/10.1007/11813040_13

"Formal Verification and Validation of ERTMS Industrial Railway Train Spacing System"
https://link.springer.com/chapter/10.1007/978-3-642-31424-7_29

_____ Evaluation Part _____

Development Functionalities

How the framework supports the construction and refinement of specification models, their translation onto executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Textualinport**

NuSMV is essentilly a model checking engine.

NuSMV textual models are edited outside the NuSMV framework, often as translations from other specification/design languages.

NuSMV is essentially a verification engine, not a design framework.

Code Generation: **No**

Document / Report Generation: **No**

Requirements Tracebiity: **No**

Model Refinement: **No**

Verification Functionalities

The approaches used by the framework to verify the models and the kind of properties that the frameworks allows to verify.

Simulation: **Textual**

Formal Verification: **Model Checking (Invariants, Linear, Branching)**

Supports both linear and branching time properties. Supports both BDD based and SMT based verification techniques. Allows to specifications of Fairness Costraints.

Supported problem size: **Large**

Model Based Testing: **No**

TimeRelatedAspects: **Yes**

Time-bounded (based step counts) versions of CTL/LTL operators are provided, in the style of TCTL

ProbabilityAspects: **No**

Property Specification Language: **LTL CTL, RTCTL, PSL**

Language Expressiveness

The characteristics of the models that can be generated within the framework.

Name of Language: **NuSMV**

Nondeterminism: **Internal / External**

Nondetermism is introduced by explicit "input variables" and by nondetermistic system initializations and trasformations.

Concurrency: **Synch**

Previous versions allowed to make use of an explicit "process" construct (now deprecated).
Data flow oriented specifications actually describe fully parallel and synchronous transformation rules.

Temporal aspects: **No**

Stochastic aspects: **No**

Modularity aspects: **Medium**

A system can be decomposed into a set of modules when global state and the global transition relation can be split in orthogonal fragments.

Supported Data Structures: **Basic**

Expression are of basic types (integer, booleans, enumerations).

Statically sized array variables are allowed.

nuXmv supports also rational numbers.

Model kind: **Logical**

Flexibility

Backword Compatibility: **Lilkely**

The modelling language is a de facto standard and quite stable. The tools is open source.

Processes and deprecated and no more supported in nuXmv.

Standard Input Format:

Import/Export to other tools: **Medium**

In the literature are founde many cases of covertions from /to nuSMV

statespace can be exported as FSM?

Modularity of the tool: **No**

The framework supports various kind of model checkers,

Team Support: **No**

Maturity

Industrial Diffusion: **Medium**

many cases of industrial uses

Stage of Development: **Yes**

Usability

Availability of Customer Support: **Partial**

It is possible tu submit bug reports (http://nusmv.fbk.eu/bug_report.html) and subscribe to mail lists for news, updates and contact with other users (nuxmv-users@list.fbk.eu).

Graphical User Interface: **No**

Easy of Use: **Medium**

Properties are encoded as LTL /CTL /PSL formulae, possibly with fairness constrains.

Quality of Documentation: **Good**

Company Constraints

Cost: **Free**

http://nusmv.fbk.eu/open_nusmv/flier.html

Supported Platfoms: **macOS, Windows, Linux**

Easyand Flexible License Management: **Easy**

Open Source license LGPL v2.1.
This license kind allows free academic and commercial usage of NuSMV. No need of further kind of licenses.

Easy to Install: **Yes**

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Medium**

6 ProB

_____Information Part_____

Tool/Framework Name: ProB

ProB is an animator, constraint solver and model checker for the B-Method (see the B-Method site of Clearsy - http://www.methode-b.com/en/). It allows fully automatic animation of B specifications, and can be used to systematically check a specification for a wide range of errors. The constraint-solving capabilities of ProB can also be used for model finding, deadlock checking and test-case generation.

The B language is rooted in predicate logic, arithmetic and set theory and provides support for data structures such as (higher-order) relations, functions and sequences. In addition to the B language, ProB also supports Event-B, CSP-M, TLA+, and Z. ProB can be installed within Rodin, where it comes with BMotionStudio to easily generate domain specific graphical visualizations. (See for an overview of ProB's components).

Commercial support is provided by the spin-off company Formal Mind (http://formalmind.com)

Pro B exists as a standalone tool or as a plugin for Rodin.

In this evaluation sheet we report the evaluation of the tool when used with its reference language, i.e. EventB. Certain observations, like no concurrency, no temporal aspects, low modularity, are strictly related to the characteristic of the B notation and would not apply when models are imported from other notiations like CSPm.

Web Sites:

http://formalmind.com
https://www3.hhu.de/stups/prob/index.php/Main_Page

Documentation:

https://www3.hhu.de/stups/prob/index.php/Documentation
http://www.atelierb.eu/wp-content/uploads/sites/3/ressources/manrefb1.8.6.uk.pdf
https://www3.hhu.de/stups/prob/index.php/User_Manual
https://www3.hhu.de/stups/prob/index.php/Tutorial
https://www3.hhu.de/stups/prob/index.php/Links
https://www3.hhu.de/stups/prob/index.php/ProB_Validation_Methods

Clearsy: "B Langauge reference Manual"
(http://www.atelierb.eu/wp-content/uploads/sites/3/ressources/manrefb1.8.6.uk.pdf)

Book "Formal Methods Applied to Complex Systems - Implementation of the B Method"
Book: J R Abrial "The B Book"

Reports on Industrial Uses of the Tool (in Railways):

"Automated Property Verification for Large Scale B Models"
https://link.springer.com/chapter/10.1007/978-3-642-05089-3_45

"Formal Implementation of Data Validation for Railway Safety-Related Systems with OVADO"
"https://link.springer.com/chapter/10.1007/978-3-319-05032-4_17"

_____ Evaluation Part _____

Development Functionalities

How the framework supports the construction and refinement of specification models, their translation onto executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Textual**

The ProB application allows the direct editing of textual models.

Code Generation: **No**

ProB does not directly support code generation. However, models can be exported to AtelierB, that does support code generation.

Document / Report Generation: **Partial**

ProB allows to visualize "statestate projections", and produce animations of the system behavior.

Requirements Tracebiity: **No**

Formal Mind distributes also the open source ProR and RIF/ReqIF tools for requirements management.

Model Refinement: **Yes**

Verification Functionalities

The approaches used by the framework to verify the models and the kind of properties that the frameworks allows to verify.

Simulation: **Textual, Graphic**

Formal Verification: **Model Checking (Invariants, Linear, Branching), Theorem Proving, Refinement Checking**

ProB offers several alternative approaches to formal verification.
See https://www3.hhu.de/stups/prob/index.php/ProB_Validation_Methods for summarising schema.

Consistency Checking (see https://www3.hhu.de/stups/prob/index.php/Consistency_Checking)
Constraint Based Checking (see https://www3.hhu.de/stups/prob/index.php/Constraint_Based_Checking)
Refinement Checking (see https://www3.hhu.de/stups/prob/index.php/Refinement_Checking)
LTL/CTL Model Checking (see https://www3.hhu.de/stups/prob/index.php/LTL_Model_Checking)
LTL Bounded Model Checking (see https://www3.hhu.de/stups/prob/index.php/Bounded_Model_Checking)

Supported problem size: **Medium**

Model Based Testing: **TestGeneration**

The framework supports the automatic generation of test
(see https://www3.hhu.de/stups/prob/index.php/Test_Case_Generation)

TimeRelatedAspects: **No**

ProbabilityAspects: **No**

Property Specification Language: **LTL, CTL,**

The supported logics are basically state based, but allow also a restricted form of event related aspects.

Language Expressiveness

The characteristics of the models that can be generated within the framework.

Name of Language: **Event B (ProB flavour)**

Nondeterminism: Internl, **External**

Incoming events constitute a possible way to introduce determinism in the system.

"CHOICE", "SELECT", "ANY" operators allows internal nondeterministic behaviors.

Concurrency: **No**

A B model can be composed by a set of elements, but the overall behavior is that one of single state machine. Imported CSPm models would not suffer of this limitation.

Temporal aspects: **No**

Stochastic aspects: **No**

Modularity aspects: **Medium**

The model is essentially a single sequential state machine. Limited forms of machine decomposition are allowed.

Supported Data Structures: **Complex**

Model kind: **Imperative**

Flexibility

Backword Compatibility: **Lilkely**

Event B is a rather standard modelling language, even if ProB adopts its own synctactic flavour.

The tool is open source and previous versions of the tool are still available for download (see https://www3.hhu.de/stups/prob/index.php/DownloadPriorVersions)

Standard Input Format: **Yes**

Import/Export to other tools: **High**

The tools also imports and verifies models in TLA+, Z, CSPm.

Modularity of the tool: **No**

The tool addresses mainly the abstract design phase of the development process

Team Support: **No**

Maturity

Industrial Diffusion: **Railway**

The Event-B language/method is widely known and used also in the railway industry.

Stage of Development: **Yes**

Actually not a long history of versions, but very stable.

Usability

Availability of Customer Support: **Yes**

Commercially provided by Formal Mind (http://formalmind.com/services/).

Graphical User Interface: **Partial**

A GUI exist. But design functionalities are limited.

Easy of Use: **Medium**

Properties are encoded as LTL / CTL formulae.

Quality of Documentation: **Good**

Company Constraints

Cost: **Free**

https://www3.hhu.de/stups/prob/index.php/ProBLicense

Supported Platfoms: **macOS, Windows, Linux**

Easy and Flexiblr License Management: : **Easy**

Open Source (see https://www3.hhu.de/stups/prob/index.php/Download

Easy to Install: : **Yes**

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Yes**

B Method is explicitly mentioned in the CENELEC norm and there is an abundant amount of literature dicumenting railway specific success stories using the method

7 AtelierB

_____Information Part_____

Tool/Framework Name: Atelier B (Rodin version)

Description:

Developed by ClearSy, Atelier B is an industrial tool that allows for the operational use of the B Method to develop defect-free proven software (formal software). It is available in 2 versions :

1- Community Edition available to anyone without any restriction,
2- Maintenance Edition for maintenance contract holders only.
3- RODIN Plugin

It is used to develop safety automatisms for the various subways installed throughout the world by Alstom and Siemens, and also for Common Criteria certification and the development of system models by ATMEL and STMicroelectronics.

Additionally, it has been used in a number of other sectors, such as the automotive industry, to model operational principles for the onboard electronics of three car models. Atelier B is also used in the aeronautics and aerospace sectors.

Atelier B exists as a standalone tool or as a plugin for Rodin.

Web Sites:

http://www.atelierb.eu/en/
http://www.clearsy.com/en/our-tools/atelier-b/
http://www.atelierb.eu/en/atelier-b-tools/
http://www.clearsy.com/en/

Documentation:

Clearsy: "B Language reference Manual"
(http://www.atelierb.eu/wp-content/uploads/sites/3/ressources/manrefb1.8.6.uk.pdf)

Book: "Formal Methods Applied to Complex Systems - Implementation of the B Method"
Book: J R Abrial "The B Book"

Reports on Industrial Uses of the Tool (in Railways):

"Safety Analysis of a CBTC System: A Rigorous Approach with Event-B"
https://link.springer.com/chapter/10.1007/978-3-319-68499-4_10

"Safe and Reliable Metro Platform Screen Doors Control/Command Systems"
https://link.springer.com/chapter/10.1007%2F978-3-540-68237-0_32

"Automated Property Verification for Large Scale B Models"
https://link.springer.com/chapter/10.1007/978-3-642-05089-3_45

"Using Formal Proof and B Method at System Level for Industrial Projects"
https://link.springer.com/chapter/10.1007/978-3-319-33951-1_2

"Aligning SysML with the B Method to Provide V&V for Systems Engineering"
https://hal.inria.fr/hal-00741134/document

"Safe and Reliable Metro Platform Screen Doors Control/Command Systems"
https://link.springer.com/chapter/10.1007%2F978-3-540-68237-0_32

_____ Evaluation Part _____

Development Functionalities

How the framework supports the construction and refinement of specification models, their translation onto executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Translators of B to C, ADA and High Integrity ADA (industrial)

Specification/Modeling: **Textual**

The specification of components is defined textually (from the tool GUI).

Code Generation: **Yes**

Translators are available from B to C, ADA and High Integrity ADA.

The documentation on the translators can be found at the url:
http://www.atelierb.eu/wp-content/uploads/sites/3/ressources/DOC/english/translators-user-manual.pdf
The free (academic license) AtelierB.app application doe not seem to contain these functionalities.

Document / Report Generation: **Partial**

The tool supports a graphical representation at the level of a project. The project components are displayed. The user can choose different display options, for example the type of links to be viewed, the view of the whole dependence graph of a project or the dependence graph of a component.

Requirements Tracebiity: **No**

Model Refinement: **Yes**

Verification Functionalities

The approaches used by the framework to verify the models and the kind of properties that the frameworks allows to verify.

Simulation: **No**

Formal Verification: **TheoremProving, Refinement Checking**

The system generates automatic proof obligations. A B component is correct when its proof obligations are demonstrated. Proofs cab be carried on in an automatic or interactive way.

Supported problem size: **Medium**

Model Based Testing: **No**

TimeRelatedAspects: **No**

ProbabilityAspects: **No**

Property Specification Language:

Language Expressiveness

The characteristics of the models that can be generated within the framework.

Name of Language: **Event B (Atelier B flavour)**

Nondeterminism: **Internal**, **External**

Incoming events constitute a possible way to introduce determinism in the system.

"CHOICE", "SELECT", "ANY" operators allows internal nondeterministic behaviors.

Concurrency: **No**

A model can be composed by a set of elements, but the overall behavior is that one of single state machine.

Temporal aspects: **No**

Stochastic aspects: **No**

Modularity aspects: **High**

Supported Data Structures: **Complex**

Model kind: **Imperative**

Flexibility

Backword Compatibility: **Moderate**

Event B is a rather standard modelling language, even if AtelierB adopts its own synctactic flavour.

Being based on the Eclipse environment, may suffer compatibility problems inherited from it.

Previous versions of the toll may be avialable under certain conditions (see http://www.atelierb.eu/en/download/.)

Standard Input Format: **Yes**

Event B specification are encoded with a tool based flavour (Atelier B). Translators exist amongthe various Ebent B flavours.

Import/Export to other tools: **Medium**

Models can be exported to Rodin/ProB-

Modularity of the tool: **Yes**

The tool is monolitic, but can be used in the abstract design, detatiled design and coding phases.

Team Support: **Yes**

Atelier B can be used by several users in a network. These users can work on the same project at the same time (see http://www.atelierb.eu/en/atelier-b-tools).

Maturity

Industrial Diffusion: **Railway**

The Event-B language/method is widely known and used also in the railway industry.

Stage of Development: **Yes**

Usability

Availability of Customer Support: **Yes**

provided by Clearsy for its maintenance edition

Graphical User Interface: **Yes**

Easy of Use: **Advanced**

Property verification may required advanced theorem proving techniques.

Quality of Documentation: **Excellent**

Company Constraints

Cost: **Free**

http://www.atelierb.eu/en/download/

The community edition is free (does not include code generators).

The maintenance edition is commercial and its price depends on the number of licenses.
(see http://www.atelierb.eu/wp-content/uploads/sites/3/atelierb/licenses/4.0/license-atelier-b-utilisation-en-V4.pdf)

http://www.atelierb.eu/en/download/distribution-policy/
http://www.atelierb.eu/en/2017/01/13/the-new-version-4-4-2-of-the-atelierb/

Supported Platfoms: **macOS, Windows, Linux, Solaris**

Easy and Flexible License Management: **Easy**

Zero cost license for the community edition (withoput support)

Easy to Install: **Yes**

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Yes**

B Method is explicitly mentioned in the CENELEC norm and there is an abundant amount of literature dicumenting railway specific success stories using the method

8 UPPAAL

_____Information Part_____

Tool/Framework Name: Uppaal

Description:

Uppaal is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types.

It is appropriate for systems that can be modeled as a collection of non-deterministic processes with finite control structure and real-valued clocks, communicating through channels or shared variables. Typical application areas include real-time controllers

and communication protocols in particular, those where timing aspects are critical.

The tool is developed in collaboration between the Department of Information Technology at Uppsala University, Sweden and the Department of Computer Science at Aalborg University in Denmark.

Web Sites:

http://www.uppaal.org/
http://www.uppaal.com/

Documentation:

http://www.it.uu.se/research/group/darts/uppaal/documentation.shtml
http://www.it.uu.se/research/group/darts/papers/texts/uppaal-smc-tutorial.pdf
http://www.it.uu.se/research/group/darts/uppaal/small_tutorial.pdf
http://www.it.uu.se/research/group/darts/papers/texts/new-tutorial.pdf

Reports on Industrial Uses of the Tool (in Railways):

"Verification and Implementation of the Protocol Standard in Train Control System"
http://ieeexplore.ieee.org/document/6649879/

_____ Evaluation Part _____

Development Functionalities

How the framework supports the construction and refinement of specification models, their translation onto executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Graphical**

Code Generation: **No**

Document / Report Generation: **Partial**

The tools allows the visualization of sequence diagrams corresponding to selected interactive simulations or counterexample guided execution paths.

Requirements Tracebiity: **No**

Model Refinement: **No**

Verification Functionalities

The approaches used by the framework to verify the models and the kind of properties that the frameworks allows to verify. The model-checker can check invariant and reachability properties by exploring the state-space of a system, i.e. reachability analysis in terms of symbolic states represented by constraints.

Simulation: **Graphic**

Formal Verification: **Model Checking (Invariants, Linear)**

The supported logic has the structure of an LTL fragment, but includes operators for reasoning about time and probabilities.

Supported problem size: **Medium**

Model Based Testing: **No**

TimeRelatedAspects: **Yes**

ProbabilityAspects: **Yes**

Property Specification Language: **MITL**

Language Expressiveness

The characteristics of the models that can be generated within the framework.

Name of Language: **Timed Automatons**

Nondeterminism: **Internal**

Concurrency: **Sync**

The model can be consitituted by a set of elements, each one with its own clock, but time advances consistently for all elements.

Temporal aspects: **Yes**

Stochastic aspects: **Yes**

Modularity aspects: **Medium**

The system can be structured into fixed set of processes.

Supported Data Structures: **Complex**

Model kind: **algebraic, graphic**

When using the graphical GUI the elements are defined in a graphical way.

The underlying code is based on the algebraic notion of timed automatons, and models can be textually encoded and verified by the command line version of the tool.

Flexibility

Backword Compatibility: **Lilkely**

UPPAAL 4.x changes the language syntax but old versions of the models are sill supported via an option (http://people.cs.aau.dk/~adavid/utap/syntax.html)

Some conversion tools ("convert.jar") are provided.

Standard Input Format: **Partial**

Import/Export to other tools: **Low**

Can import timed automaton in textual format.

Modularity of the tool: **Yes**

Team Support: **No**

Maturity

Industrial Diffusion: **Medium**

The tool is widely used in industry with several applications in the railway field.

Stage of Development: **Yes**

Usability

Availability of Customer Support: **Yes**

Provided by www.uppaal.com

Graphical User Interface: **Yes**

Easy of Use: **Medium**

System preperties are specified with a custom MITL temporal logics.

Quality of Documentation: **Good**

Company Constraints

Cost: **Mix**

Free for academic uses. Commercial licenses available.

Supported Platfoms: **macOS, Windows, Linux**

Easyand Flexible License Management: **Moderate**

Easy to Install: **Yes**

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Medium**

9 SCADE

_____Information Part_____

Tool/Framework Name: SCADE

Description:

SCADE Suite is a product line in the ANSYS embedded software family that provides a model-based development environment for critical embedded software. With native integration of the formally defined Scade language, SCADE Suite is the integrated design environment for critical applications including requirements management, model-based design, simulation, verification, qualifiable/certified code generation and interoperability with other development tools and platforms.

Web Sites:

https://www.ansys.com/products/embedded-software/ansys-scade-suite
http://www.esterel-technologies.com/

Documentation:

https://www.ansys.com/-/media/ansys/corporate/resourcelibrary/brochure/tds-ansys-scade-suite-17.pdf
http://www.esterel-technologies.com/industries/en-50128/

Reports on Industrial Uses of the Tool (in Railways):

"Modeling and verification of zone controller: the scade experience in China's railway systems"
https://dl.acm.org/citation.cfm?id=2819430

"A Parametric Dataflow Model for the Speed and Distance Monitoring in Novel Train Control Systems"

https://link.springer.com/chapter/10.1007/978-3-319-25141-7_5

_____ Evaluation Part _____

Development Functionalities

How the framework supports the construction and refinement of specification models, their translation onto executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Graphical**

Code Generation: **Yes**

Document / Report Generation: **Yes**

Requirements Tracebiity: **Yes**

Model Refinement: **Yes**

Verification Functionalities

The approaches used by the framework to verify the models and the kind of properties that the frameworks allows to verify.

Simulation: **Graphical**

Formal Verification: **Model Checking (Invariants)**

Supported problem size: **Limited**

Model Based Testing: **TestGeneration**

TimeRelatedAspects: **No**

ProbabilityAspects: **No**

Property Specification Language: **SCADE**

Language Expressiveness

The characteristics of the models that can be generated within the framework.

Name of Language: **SCADE**

Nondeterminism: **External**

Concurrency: **Sync**

Temporal aspects: **Yes**

Stochastic aspects: **No**

Modularity aspects: **High**

Supported Data Structures: **Complex**

Model kind: **graphic**

Flexibility

Backword Compatibility: **Likely**

It is a company policy to continuously support previous versions of the tools for the clients.

(private communication)

Standard Input Format: **Partial**

The models are saved in etp textual format.

Import/Export to other tools: **Low**

The tools can import Simulink models. Third parties tools allow to import SCADE models.

But overall the tool is not oriented to being open.

Modularity of the tool: **Yes**

Team Support: **No**

Maturity

Industrial Diffusion: **Railway**

SCADE is widely known and used also in the railway industry.

Stage of Development: **Yes**

Usability

Availability of Customer Support: **Yes**

With commercial license.

Graphical User Interface: **Yes**

Methematical Skills: **Basic**

System properties are specified by graphically combining predefined operators.

No deep mathematical knowledges are needed.

Quality of Documentation: **Excellent**

Only for clients.

Company Constraints

Cost: **Pay**

Supported Platfoms: **Windows**

Easyand Flexible License Management: **Complex**

Student, academic, commercial, based on the set of functionalities required.

Easy to Install: **Yes**

Railway Specific Criteria

CENELEC certification: **Yes**

see http://www.esterel-technologies.com/industries/en-50128/

Integration into the CENELEC process: **Yes**

10 FDR4

_____Information Part_____

Tool/Framework Name: FDR4

Description:

FDR4 is a refinement checker that allows the user to verify properties of programs written in CSPM, a language that combines the operators of Hoare's CSP with a functional programming language. Originally developed by Formal Systems (Europe) Ltd in 2001, since 2008 is supported by the Computer Sceince Department of University of Oxford.

Web Sites:

https://www.cs.ox.ac.uk/projects/fdr/

Documentation:

https://www.cs.ox.ac.uk/projects/fdr/manual/

"The Theory and Practice of Concurrency"
https://www.cs.ox.ac.uk/bill.roscoe/publications/68b.pdf

Reports on Industrial Uses of the Tool (in Railways):

"A formal specification of an automatic train protection system"
https://link.springer.com/chapter/10.1007/3-540-58555-9_118

_____ Evaluation Part _____

Development Functionalities

How the framework supports the construction and refinement of specification models, their translation onto executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Textualinport**

The tool just operates textual pre-existing models written in CSPm.

Code Generation: **No**

Document / Report Generation: **Partial**

The tools allows the visualization of abstract views of the model behavior (e.g. by hiding not relevant transitions and minimizing the graph according to selected equivalence relations).

Requirements Tracebiity: **No**

Model Refinement: **Yes**

Verification Functionalities

The approaches used by the framework to verify the models and the kind of properties that the frameworks allows to verify.

Simulation: **Textual**

The possible evolutions of a process can be probed interactively (optionally Guided, interactive, Random)

Formal Verification: **RefChecking**

Supported problem size: **Large**

Model Based Testing: **No**

TimeRelatedAspects: **Yes**

ProbabilityAspects: **No**

Property Specification Language: **n/a**

Language Expressiveness

The characteristics of the models that can be generated within the framework.

Name of Language: **CSPm, tock-CSP**

Nondeterminism: **Internal, External**

Concurrency: **Async**

The model is constituted by a set of concurrent processes communicating through synchronous communication channels (no shared memory among processes).

Temporal aspects: **Yes**

The tock-CSP language, for the design of timed system, is supported.

Stochastic aspects: **No**

Modularity aspects: **High**

Supported Data Structures: **Complex**

The ML functional language is used for the definition of data values and types.

Model kind: **algebraic, functional**

Flexibility

Backword Compatibility: **Moderate**

The modelling language is rather stable and standard but he tolls is not open source, and little evidence of attention to backward compatibility issues has been found.
(https://www.cs.ox.ac.uk/projects/fdr/manual/changes.html)

Standard Input Format: **Yes**

CSPm and tock-CSP are the standard language references

Import/Export to other tools: **Medium**

Language is open and not proprietary. Indeed ProB also operated onCSPm models complementing its functionalities.

Modularity of the tool: **No**

Team Support: **No**

Maturity

Industrial Diffusion: **Low**

Little evidence of industrial uses has been found.

Stage of Development: **Yes**

Usability

Availability of Customer Support: **Partial**

There is a mailing list for announcements and bug reports.

Graphical User Interface: **Limited**

Easy of Use: **Medium**

System properties are expressed in terms of various kinds of refinement relations.

Quality of Documentation: **Excellent**

Company Constraints

Cost: **Mix**

FDR is only freely available for academic teaching and research purposes.

For commercial /evaluation licenses is given the contact: fdr-queries@cs.ox.ac.uk

https://www.cs.ox.ac.uk/projects/fdr/licensing.html

https://www.cs.ox.ac.uk/projects/fdr/manual/licenses.html

Supported Platfoms: **macOS,Windows,Linux**

Easyand Flexible License Management: **Moderate**

Easy to Install: **Yes**

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Medium**

CSP appears to be widely used for the design of railway systems, altough FDR4 is not explicitly mentioned in the analysed literature it is one of the few tools that support CSP designs.

11 CPN-Tools

_____Information Part_____

Tool/Framework Name: CPN Tools

Description:

CPN Tools is an environment for editing, simulating, and analysing Colored Petri Nets. It is originally developed by the CPN Group at Aarhus University from 2000 to 2010. The main architects behind the tool are Kurt Jensen, Søren Christensen, Lars M. Kristensen, and Michael Westergaard. From the autumn of 2010, CPN Tools is transferred to the AIS group, Eindhoven University of Technology, The Netherlands.

Web Sites:

http://cpntools.org/

http://sml-family.org/

Documentation:

http://cpntools.org/2018/01/16/documentation-2/
http://cpntools.org/2018/01/16/state-space-analysis-2/
Book: Coloured Petri Nets — Modeling and Validation of Concurrent Systems.

Reports on Industrial Uses of the Tool (in Railways):

"Model-based test generation techniques verifying the on-board module of a satellite-based train control system model"
http://ieeexplore.ieee.org/document/6696307/

_____ Evaluation Part _____

Development Functionalities

How the framework supports the construction and refinement of specification models, their translation onto executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Graphical**

Code Generation: **No**

Yes: The tool supports the automatic generation of program code from the models.

Document / Report Generation: **No**

Requirements Tracebiity: **No**

Model Refinement: **No**

Verification Functionalities

The approaches used by the framework to verify the models and the kind of properties that the frameworks allows to verify.

Simulation: **Graphical**

Formal Verification: **Model Checking (Invariants, Branching)**

Supported problem size: **Limited**

Model Based Testing: **No**

TimeRelatedAspects: **Yes**

ProbabilityAspects: **No**

Property Specification Language: **CPN**

Language Expressiveness

The characteristics of the models that can be generated within the framework.

Name of Language: **CPN**

Nondeterminism: **Internal**

Concurrency: **Async**

Temporal aspects: **Yes**

Stochastic aspects: **No**

Modularity aspects: **High**

Supported Data Structures: **Complex**

Data types, data values and auxiliary function are defined in the functional language "Standard ML"

Model kind: **functional, graphic**

Flexibility

Backword Compatibility: **Likely**

The sources of the various components of the framework are available (current and old versions, see http://cpntools.org/2018/01/15/source/).

There are some backward compatibility issues of new versions of the tools w.r.t old versions of the models.

(see http://cpntools.org/2018/01/23/change-logs/).

The issue of backaward compatibility is however taken into due consideration by the developers.

Standard Input Format: **Partial**

Import/Export to other tools: **Medium**

Modularity of the tool: **No**

Team Support: **No**

Maturity

Industrial Diffusion: **Medium**

There is a list of industrial projects using CP nets, some of them are railway related.

(see http://cs.au.dk/cpnets/industrial-use/)

Stage of Development: **Yes**

Usability

Availability of Customer Support: **Partial**

Graphical User Interface: **Partial**

Easy of Use: **Medium**

System properties are encoded as logial CTL formulae.

Quality of Documentation: **Good**

Company Constraints

Cost: **Pay**

http://cpntools.org/category/licenses/

Supported Platfoms: **Windows**

Easyand Flexible License Management: **Easy**

The CPN Tools GUI is licensed under the GNU General Public License (GPL) version 2.

Easy to Install: **Yes**

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Medium**

12 CADP

_____Information Part_____

Tool/Framework Name: CADP

Description:

CADP (Construction and Analysis of Distributed Processes) is a verification framework for the design of asynchronous concurrent systems. While its origins dates back to the mid 80s, since than it has been continuously improved and enriched, and is currently actively maintained by the CONVECS team at INRIA.

Web Sites:

http://cadp.inria.fr/

Documentation:

http://cadp.inria.fr/tutorial/
http://cadp.inria.fr/man/
http://cadp.inria.fr/publications/
http://cadp.inria.fr/tools.html

Reports on Industrial Uses of the Tool (in Railways):

_____ Evaluation Part _____

Development Functionalities

How the framework supports the construction and refinement of specification models, their translation onto executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Textualinport**

LNT or LOTOS models are generated in plain textual form outside of the framework.

Code Generation: **Yes**

The tool "ceasar.adt" allows the translation of a LOTOS process into an executable "C" program. Specifications in the "LNT" language are translated into LOTOS by the "LNT.open" tool.

Document / Report Generation: **Partial**

The tools allows the visualization of abstract views of the model behavior (e.g. by hiding not relevant transitions and minimizing the graph according to selected equivalence relations).

Requirements Tracebiity: **No**

Model Refinement: **No**

Verification Functionalities

The approaches used by the framework to verify the models and the kind of properties that the frameworks allows to verify.

Simulation: **Textual**

Formal Verification: **Model Checking (Invariants, Branching)**

Several verification engines are provided ("evaluator3", "evaluator4") that allow the on the fly verification of system properties expressed in MCF, through their translation into a boolean equation system (BES). Further verification functionalities are provided by the "bcg_min" tool that allow the generation of abstract minimizations of a system according to several predefined equivalence relations.

Supported problem size: **Large**

The supported size of "bcg" graph (explicilty modelling the evolutions of a system) is in the order of the millions of states. However the overall system behavior can be described by a concurrent set of "bcg" graphs, each of which can represent a minimised version of the corresponding system component. This compositional apporoach to verifications allows the compositional analysis of large complex systems.

Model Based Testing: **Yes**

see JTorX, tgv

TimeRelatedAspects: **No**

ProbabilityAspects: **No**

Property Specification Language: **MCL, XTL**

MCL is an extension of alternation free mu-calculus with regular expressions and parametric fix points

Language Expressiveness

The characteristics of the models that can be generated within the framework.

Name of Language: **LOTOS, LNT**

LOTOS and LNT are the two languages natively supported

Nondeterminism: **Internal, External**

Concurrency: **Async**

The model is constituted by a set of concurrent processes communicating through synchronous communication channels (no shared memory among processes).

Temporal aspects: **No**

Stochastic aspects: **No**

Modularity aspects: **High**

Yes: The tool allows the user to model in a hierarchical way, and the partitioning of the model Into modules

Supported Data Structures: **Complex**

Model kind: **Imperative, algebraic**

LNT is a language with an imperative style, LOTOS is a process algebraic language.

Flexibility

Backword Compatibility: **Likely**

Even if the reference language (LOTOS) is an ISO standard, the tool is not open source, and in some cases backward compatibility issues may raise (see http://cadp.inria.fr/changes.html). The framework provides conversion aids for specific incompatibility issues.

Standard Input Format: **Yes**

Import/Export to other tools: **High**

The tool allows the user to import and export LTS in various formats.

Modularity of the tool: **Yes**

The CADP framework is a collection of more then 50 programs performing various kind of activities.

Team Support: **No**

Maturity

Industrial Diffusion: **Medium**

Widely in several industrial industrial projects (especially in the communications / hardware circuits fields), but rarely adopted in railway related projects.

(see http://cadp.inria.fr/case-studies/)

Stage of Development: **Yes**

Usability

Availability of Customer Support: **Partial**

Graphical User Interface: **Limited**
See the eucaliptus tool (xeuca)

Easy of Use: **Adavnced**

System Properties are expressed in terms or alternation free mu-calculus formulae, and compositional minimizations performed according to several LTL bisimulation strategies.

Quality of Documentation: **Good**

A lot of documentation is available in the form of manual pages, articles, tutorial, books, but the search of all this documentation for the search of specific queries is not immediate.

Company Constraints

Cost: **Mix**

Free for all or for academic uses, commercial licenses available

Supported Platfoms: **macOS, Windows, Linux**

Flexible Licenses: **Yes**

Easy License Management: **Moderate**
The license is bound to a specific machine(s) and must be renewed every year.

Easy to Install: **Partial**

The framework has several dependences to other software components. (e.g. X11,

Developers Tools with Command Line extensions, Postscript Viewers, gnutar, wget) and

has a not trivial installation procedure.

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Medium**

The Language LOTOS (ISO standard) is explicitly mentioned, among all the other formal languages and techniques, in the CENELEC norm (pag 104, D28.4). The tool is rarely used in the railway field.

13 mCRL2

_____Information Part_____

Tool/Framework Name: mCRL2

Description:

mCRL is a formal specification language with an associated toolset. The toolset can be used for modelling, validation and verification of concurrent systems and proto- cols. The mCRL2 toolset is developed at the department of Mathematics and Computer Science of the Technische Uni- versiteit Eindhoven, in collaboration with LaQuSo, CWI and the University of Twente. The mCRL2 language is based on the Algebra of Communicating Processes (ACP) which is extended to include data and time.

Web Sites:

http://mcrl2.org/web/user_manual/index.html

Documentation:

http://mcrl2.org/web/user_manual/user.html
http://www.mcrl2.org/web/user_manual/language_reference/mucalc.html

Reports on Industrial Uses of the Tool (in Railways):

_____ Evaluation Part _____

Development Functionalities

How the framework supports the construction and refinement of specification models, their translation onto executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Textual**

Pre-existing textual models can be selected from the "mCRL2.app" GUI, or edited with the "mcrl2xi" program.

Code Generation: **No**

Document / Report Generation: **Partial**

The tools allows the visualization of abstract views of the model behavior (e.g. by hiding not relevant transitions and minimizing the graph according to selected equivalence relations).

Requirements Tracebility: **No**

Model Refinement: **No**

Verification Functionalities

The approaches used by the framework to verify the models and the kind of properties that the frameworks allows to verify.

Simulation: **Textual**

Simulation can be interactively monitored with the functionalities providd by LpsXsim

Formal Verification: **Model Checking (Invariants, Branching)**

The login supported is the mu-calculus extended with patter matching operators, time operators, and parametric fix points. Once a system has been translated into an explicit "lts", various minimisatoins and equivalence checking features become available.

Supported problem size: **Medium**

Model Based Testing: **No**

TimeRelatedAspects: **Yes**

ProbabilityAspects: **No**

Property Specification Language:

mu-calculus extended with regular expressions.

Language Expressiveness

The characteristics of the models that can be generated within the framework.

Name of Language: **mCRL2**

(just informative aspect)

Nondeterminism: **Internal, External**

An explicit nondeterministic choice operator models alternative behaviours.

Concurrency: **Async**

The model is constituted by a set of concurrent processes communicating through synchronous communication channels (no shared memory among processes).

Temporal aspects: **Yes**

Stochastic aspects: **No**

Modularity aspects: **High**

Supported Data Structures: **Complex**

Model kind: **algebraic**

Flexibility

Backword Compatibility: **Lilkely**

The tool is open source, and old version are available. The modelling language is rather stable and standard de facto. (se http://mcrl2.org/web/user_manual/historic_releases.html#historic-releases)

Standard Input Format: **Yes**

Import/Export to other tools: **High**

Models in mcrl2 textual format can be converted first in lps format and then in lts format. from the lts format can be converted into the "aut" (CADP), "dot" (GRAPHVIZ"), fsm (Finite State MAchine) format.

Modularity of the tool: **Yes**

The mCRL2 framework is a collection of more then 50 programs performing various kind of activities.

Team Support: **Yes / No**

Yes: The tool supports collaborative team development

Maturity

Industrial Diffusion: **Low**

Appears to be used in some industrial projects, but not in railway related projects.

(see http://mcrl2.org/web/user_manual/showcases.html,

http://mcrl2.org/web/user_manual/publications.html

Stage of Development: **Yes**

Usability

Availability of Customer Support: **Partial**

Graphical User Interface: **Limited**

Easy of Use: **Adanced**

System Properties are expressed in terms or parametric mu-calculus formulae.

System minimizations/ equivalence checking can be performed according to several LTL bisimulation strategies.

Quality of Documentation: **Good**

Company Constraints

Cost: **Free**

Supported Platfoms: **macOS, Windows, Linux**

Easyand Flexible License Management: **Easy**

Easy to Install: **Partial**

Requires Developers Tools with Command Line extensions (e.g. gcc compilation system) when some options are used.

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Low**

14 SAL

_____Information Part_____

Tool/Framework Name: SAL

Description:

SAL stands for Symbolic Analysis Laboratory. It is a framework for combining different tools for abstraction, program analysis, theorem proving, and model checking toward the calculation of properties (symbolic analysis) of transition systems. A key part of the SAL framework is an intermediate language, developed in collaboration with Stanford and Berkeley, for describing transition systems and specifying concurrent systems in a compositional way. This language is intended to serve as the target for translators that extract the transition system description for other modeling and programming languages, and as a common source for driving different analysis tools. It is supported by a tool suite that includes state of the art symbolic (BDD-based) and bounded (SAT-based) model checkers, an experimental "Witness" model checker, and a unique "infinite" bounded model checker based on SMT solving. Auxiliary tools include a simulator, deadlock checker and an automated test generator.

Web Sites:

http://sal.csl.sri.com

Documentation:

http://fm.csl.sri.com/
http://sal.csl.sri.com/documentation.shtml
http://sal.csl.sri.com/doc/language-report.pdf
http://sal.csl.sri.com/doc/salenv_tutorial.pdf
http://www.csl.sri.com/users/rushby/slides/fm-tut.pdf
http://www.csl.sri.com/users/bruno/publis/sri-sdl-04-03.pdf
http://sal.csl.sri.com/hybridsal/

Reports on Industrial Uses of the Tool (in Railways):

_____ Evaluation Part _____

Development Functionalities

How the framework supports the construction and refinement of specification models, their translation onto executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Textualinport**

SAL textual models are edited outside the SAL framework.

Code Generation: **No**

Document / Report Generation: **No**

Requirements Tracebiity: **No**

Model Refinement: **No**

Verification Functionalities

The approaches used by the framework to verify the models and the kind of properties that the frameworks allows to verify.

Simulation: **Textual**

The tools "sal-sim" allows for interactive simulations.

Formal Verification: **TheoremProving, Model Checking (Invariant, Linear)**

Explict model checking ("sal-esmc"), BDD based Symbolic model checking("sal-smc"), Bounded SAT/SMT based model checking ("sal-bmc"), using Yices,also for infinite states systems ("sal-inf-bmc"), Theorem proving using "PVS".

(more complete info and tutorial at http://www.csl.sri.com/users/rushby/slides/fm-tut.pdf)

Supported problem size: **Large**

Model Based Testing: **TestGeneration**

TimeRelatedAspects: **No**

ProbabilityAspects: **No**

Property Specification Language: **LTL**

Language Expressiveness

The characteristics of the models that can be generated within the framework.

Name of Language: **SAL**

Nondeterminism: **Internal**, **External**

External nondeterminism provided by inputs variables, internal by nonterministic assignments.

Concurrency: **Async,Sync**

SAL modeules can be composed in a synchronous or asynchronous way.

Temporal aspects: **Yes**

for documentation of timed systems in SAL see
http://www.csl.sri.com/users/bruno/publis/sri-sdl-04-03.pdf

see also http://sal.csl.sri.com/hybridsal/ for documenttion oh hybrid systems in HubridSal.

Stochastic aspects: **No**

Modularity aspects: **Medium**

Supported Data Structures: **Complex**

Model kind: **logical**

Flexibility

Backword Compatibility: **Moderate**

The tool is Open Source, and previous versions are available (see http://sal.csl.sri.com/download.shtml), but there is no evidence of particular attention to backword compatibility issues

Standard Input Format: **Yes**

Import/Export to other tools: **Medium**

The tool defines an abstract XML syntax (SAL DTD) for its modelling language to make easier the interactions with other tools.

Modularity of the tool: **No**

The Symbolic Analysis Laboratory comprises a rich set of tools, but all related to verification aspects.

Team Support: **No**

Maturity

Industrial Diffusion: **Low**

Little evidence of indusrial uses.

Stage of Development: **Yes**

Usability

Availability of Customer Support: **Partial**

Mailing lists are available for announcements, bug notifications, and discussions (http://sal.csl.sri.com/mailing_lists.shtml)

Graphical User Interface: **No**

Easy of Use: **Advanced**

System properties can be model checked as LTL formulae, or can be verified through advanced theorem proving techniques.

Quality of Documentation: **Good**

Company Constraints

Cost: **Free**

http://sal.csl.sri.com/download.shtml

Supported Platfoms: **macOS, Windows, Linux**

Easyand Flexible License Management: **Easy**

Easy to Install: **Partial**

A few missing (not found) dynamic libraries issues have to be fixed.

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Low**

15 TLA+

_____Information Part_____

Tool/Framework Name:

Description:

The TLA Toolbox is an IDE (integrated development environment) for the TLA+ tools.

Web Sites:

https://lamport.azurewebsites.net/tla/toolbox.html

Documentation:

Reports on Industrial Uses of the Tool (in Railways):

_____ Evaluation Part _____

Development Functionalities

How the framework supports the construction and refinement of specification models, their translation onto executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Textual**

Code Generation: **No**

Document / Report Generation: **No**

Requirements Tracebiity: **No**

Model Refinement: **No**

Verification Functionalities

The approaches used by the framework to verify the models and the kind of properties that the frameworks allows to verify.

Simulation: **No**

Formal Verification: **TheoremProving/ Model Checking (nvariant, Linear)**

Supported problem size: **Medium**

Model Based Testing: **No**

TimeRelatedAspects: **No**

ProbabilityAspects: **No**

Property Specification Language: **LTL**

(just informative)

Language Expressiveness

The characteristics of the models that can be generated within the framework.

Name of Language: **name**

(just informative aspect)

Nondeterminism: **Internal**

Thye TLAPlus language provides the "with" and "either" operators to deal with nondeterminism, plus nondeterministic scheduling among processes.

External: the model allows nondeterminism associated to external inputs or trigger-events

No: the model is fully deterministic

Concurrency: **Async**

Processes are scheduled by interleaving.

Temporal aspects: **No**

Stochastic aspects: **No**

Modularity aspects: **Medium**

Supported Data Structures: **Complex**

The support data types include the basic types (numbers, strings, booleans) sets (dfined by enumeration or by properties), records , sequences and functions.

Model kind: **logic**

Flexibility

Backword Compatibility: **Moderate**

The tool is Open Source and previous releases are available.

No specific evidence about backward compatibility issues has been found.

Standard Input Format: **Yes**

Import/Export to other tools: **Low**

The tool is not particularly oriented for inport/export of models.

Modularity of the tool: **Low**

Team Support: **No**

Maturity

Industrial Diffusion: **Low**

There are not many examples of indutrial uses of TLA+, nine of which railway related

(see https://en.wikipedia.org/wiki/TLA%2B#Industry_use)

Partial: There are know cases of use of the tool in industry

Stage of Development: **Yes**

Usability

Availability of Customer Support: **Partial**

Several free access user groups sites exists:

(https://groups.google.com/forum/#!forum/tlaplus, https://www.reddit.com/r/tlaplus/)

(see also https://lamport.azurewebsites.net/tla/hyperbook.html)

Graphical User Interface: **Limited**

Easy of Use: **Advanced**

System properties can be model checked as LTL formulae, or can be verified through advanced theorem proving techniques.

Quality of Documentation: **Good**

Company Constraints

Cost: **Free**

https://lamport.azurewebsites.net/tla/license.html

Supported Platfoms: **macOS, Windows, Linux**

names of platforms: the names of the suported platforms (macOS,Windows,Linux,Solaris,...)

Easyand Flexible License Management: **Easy**

Easy to Install: **Yes**

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Low**

16 UMC

_____Information Part_____

Tool/Framework Name: KandISTI/UMC

Description:

UMC is a verification framework developed at the FM&&T Laboratory of ISTI-CNR for the definition, exploration, analysis and model checking of system designs represented as a set of communicating (UML) state machines.

Web Sites:

http://fmt.isti.cnr.it/umc

Documentation:

http://fmt.isti.cnr.it/umc/DOCS/

Reports on Industrial Uses of the Tool (in Railways):

_____ Evaluation Part _____

Development Functionalities

How the framework supports the construction and refinement of specification models, their translation onto executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Textual**

Code Generation: **No**

Document / Report Generation: **Partial**

The tool allows to generate minimised abstractions of the system behaviour.

Requirements Tracebiity: **No**

Model Refinement: **No**

Verification Functionalities

The approaches used by the framework to verify the models and the kind of properties that the frameworks allows to verify.

Simulation: **Textual**

Formal Verification: **Model Checking (Invariants, Branching)**

The supported CTL/ACTL like logic is state/event based and allows to reason both on properties of states and on properties of transitions.

Supported problem size: **Medium**

Model Based Testing: **No**

TimeRelatedAspects: **No**

ProbabilityAspects: **No**

Property Specification Language: **SocI, UCTL**

Language Expressiveness

The characteristics of the models that can be generated within the framework.

Name of Language: **UMC**

Nondeterminism: **Internal**

The model behaviour is defined by a set event/condition/action rules. The action part of he rules defines deterministioc transformation of the local state of a state machine. if several rules are applicable, any of them can be nondeterministically selected. If several state machines are ready to evolve (i.e. have fireable transition rules) any of the can be nondeterministically selected for the system evolution.

Concurrency: **Async, Synch**

The system is represented by a set of concurrent (communicating) state machines. The concurrency among machines is modelled by interleaving. A state machine can be contain composite substates composed by parallel regions that evolve synchronously.

Temporal aspects: **No**

Stochastic aspects: **No**

Modularity aspects: **High**

At the top level struacture we have just a static set of state machines, defined by statecharts. The structure of a statechart can be defined in hierarchical and modular way using composite states and concurrent regions.

Supported Data Structures: **Complex**

Etherogenous, dynamically sized arrays are supported.

Model kind: **Imperative**

Flexibility

Backword Compatibility: **Moderate**

Standard Input Format: **Yes**

Import/Export to other tools: **Low**

Modularity of the tool: **Low**

Team Support: **No**

Maturity

Industrial Diffusion: **No**

Stage of Development: **No**

The tool has a long history of versions, and is quite stable, but its development does not have the robustness of a commercial tool.

Usability

Availability of Customer Support: **Partial**

Graphical User Interface: **Partial**

Easy of Use: **Medium**

System properties can be model checked as CTL temporal logic formulae.

Quality of Documentation: **Limited**

Company Constraints

Cos: **Free**

Supported Platfoms: **macOS, Windows, Linux**

The graphic user interface is supported only by macOS.

Easy and Flexible License Management: **Easy**

Easy to Install: **Yes**

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Low**