Tool Evaluation - Technical Annex to the Paper

“Systematic Evaluation and Usability Analysis of Formal Tools for Railway System Design”

Franco Mazzanti, Alessio Ferrari, Davide Basile, Maurice ter Beek

CNR-ISTI, Via G. Moruzzi 1, Pisa, Italy

Email: {franco.mazzanti, alessio.ferrari, davide.basile, maurice.terbeek}@isti.cnr.it

Table of Contents

Table of Contents 1

Overview 2

The Evaluation Sheet Reference Template 3

SPIN 10

Simulink 14

NuSMV/nuXmv 18

ProB 22

AtelierB 26

UPPAAL 30

FDR4 33

CPN-Tools 37

CADP 40

mCRL2 44

SAL 48

TLA+ 53

UMC 56

1 Overview

This document collects the evaluation sheets of 13 tools for system design, namely CADP (2020-g), FDR4 (4.2.7), NuSMV(1.1.1), ProB(1.9.3), Atelier B (4.5.1), Simulink (R2020a), SPIN (6.4.9), UMC (4.8), UPPAAL (4.1.4), mCLR2 (202006.0), SAL (3.3), TLA+ (2) and CPN Tools (4.0).

The tools were evaluated by three assessors following the steps described below:

1) install and run the tool;

2) consult the website of the tool, to check the official documentation;

3) opportunistically search for additional documentation to identify useful information to fill the evaluation sheet;

4) refer to the structured list of papers on formal methods and railways published at https://goo.gl/TqGQx5, to check for tools' applications in railways;

5) perform some trials with the tools to confirm claims reported in the documentation, and assign the value to those features that required some hands-on activity to be evaluated;

6) report the evaluation on the sheet, together with the links to the consulted documents and papers, and appropriate notes when the motivation of some assignment needed clarification.

In the following section the reference document for assessment, with features and values, is reported. Then, for each tool, we attach the associated sheet.

2 The Evaluation Sheet Reference 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 into executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

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

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

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

Textualimport (TEXTIM): 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.

Project Management: **Yes, No**

Yes: The framework supports the management of a project and the GUI-based navigation of its conceptual components (models, submodels, verification results, tests, etc.).

No: No project management is supported.

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

Textual (TEXT): 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: the tool does not support simulation of the possible system evolutions

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

Refinement Checking (RF): Allows to verify equivalence / refinement relations among models. Note that refinement can also be proved through, e.g. theorem proving, here for refinement checking we intend automatic checking of refinement.

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).

Model Checking Observer (MC-O): The property to be verified is expressed in the same graphical language of the tool, in the form of an observer block. The model assumes inputs and checks that the output fulfils the desired value.

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

Large-scale Verification Technique: **On-the-fly model checking, Partial order reduction, Bounded Model Checking, Symbolic Model Checking, SAT-SMT Constraint Solving and Theorem Proving, Statistical Model Checking, Compositionality and Minimization**

On-the-fly model checking (FLY): the state is generated on demand.

Partial order reduction (POR): exploitation of symmetries in the state space.

Parallel computation (PAR): parallel computation distributed on more hosts.

Bounded Model Checking (BMC): state space exploration up to a certain depth.

Symbolic Model Checking (SYM): compact state space representation.

SAT-SMT Constraint Solving and Theorem Proving (SCT): avoid explicit reasoning on the state space.

Statistical Model Checking (SMC): avoid state space generation using simulations and provide an approximate solution.

Compositionality and Minimization (COM): divide the problem into smaller subproblems.

Model Based Testing: **Yes, No**

Yes: The framework supports the automatic generation of test

No: The framework does not support automatic generation of test

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 Choice (INT), External Choice (EXT)**

INT: the model allows internal non deterministic system evolutions

EXT: the model allows external choices associated to inputs or trigger-events

Concurrency: **Asynch, Synch, A/Synch, No**

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

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

A/Synch: The modelling language supports synchronous and asynchronous elements

No: The model is constituted by just one element

Timing aspects: **Yes, No**

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

No: The language does not have this feature.

Probabilistic or Stochastic aspects: **Yes, No**

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

No: the language does not have this feature.

Modularity of the Language: **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.

No: no modules supported.

Supported Data Structures: **Basic, Complex**

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

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

Float Support: **Yes, No**

Yes: The language supports floating point numbers as primitive types

No: Otherwise

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

Tool Flexibility

Backward 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 if 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: **Standard, Open, Partial**

Standard: The input language is based on a language standardised by an international organization (e.g. ISO).

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

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

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

High: The tool provides several import/export functionalities

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

(Known cases of interactions with tools are mentioned.)

Low: Tool not oriented towards export/export functionalities

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

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

Medium: The tool offers multiple functionalities, but not in the form of packages that can be loaded and combined.

Low: The tool has a limited number of functionalities in a monolithic environment.

Team Support: **Yes, No**

Yes: The tool supports collaborative team development.

No: The tool does not have this feature.

Maturity

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

High: The tool is claimed to be used in industry, and the website of the tool reports several industrial cases.

Medium: The tool is claimed to be partially used in industry, and the website of the tool reports a limited number of industrial cases.

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

Stage of Development: **Mature, Partial, Prototype**

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

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

Prototype: The tool is at the level of a prototype

Usability

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

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 need to be 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.

Mathematical Background: **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 knowledge of temporal logics

Advanced: the tool requires advanced logical/mathematical skills, such as theorem proving and process algebras.

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 (e.g., 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 - ALL if all all three are supported)

Complexity of 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 several problems were encountered with the license management system.

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.

https://www3.risc.jku.at/education/oldmoodle/file.php/9/Spin-Introduction.pdf

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 into 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 capabilities. 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 also allow 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 Traceability: **No**

Project Management: **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 (Linear)**

Large-scale Verification Technique: **On-the-fly Model Checking, Partial Order Reduction, Parallel Computation**

Model Based Testing: **No**

Property Specification Language: **LTL**

(just informative)

Notes:

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

Language Expressiveness

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

Name of Language: **Promela**

Nondeterminism: **Internal**

The choice operator allows non deterministic selection of transition rules.

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

Concurrency: **Asynch**

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

Timing 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.

http://spinroot.com/spin/Man/arrays.html

Float Support: **No**

Model kind: **Imperative**

Flexibility

Backward Compatibility: **Likely**

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

Standard Input Format: **Open**

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

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

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

Modularity of the tool: **Low**

Team Support: **No**

Maturity

Industrial Diffusion: **High**

Many cases of industrial uses.

Stage of Development: **Mature**

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.

Mathematical Background: **Medium**

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

Quality of Documentation: **Good**

Company Constraints

Cost: **Free**

Free software BSD-3Clause license

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

Supported Platforms: **ALL**

Complexity of License Management: **Easy**

Easy to Install: **Yes**

For MAC OS it 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 tool appears to be used in the railway field but without explicit references to CENELEC related activities.

Notes

4 Simulink

______________Information Part____________

Tool/Framework Name: Simulink

Description:

Simulink, developed by MathWorks, is a graphical programming environment for modeling, simulating and analyzing multi domain 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
https://www.mathworks.com/help/sldv/ug/workflow-for-proving-model-properties.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 into 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 Traceability: **Yes**

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

Project Management: **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 Observer**

The model must be extended with "observer" components to highlight the properties of interest. Although it does not use the LTL syntax to express the properties, linear time properties can be encoded in blocks.

The "Design Verifier" functionality calls a SAT based Bounded model checker.
https://it.mathworks.com/discovery/formal-verification.html
https://it.mathworks.com/products/sldesignverifier.html
https://it.mathworks.com/help/sldv/ug/workflow-for-proving-model-properties.html

Large-scale Verification Technique: **Bounded Model Checking**

Model Based Testing: **YES**

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

Property Specification Language:

Properties are specified in the Simulink language, and observers and proof operators

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.

Timing aspects: **Yes**

The Clock block represents the current simulation time, which can be retrieved also with the function getSimulationTime() in Stateflow.

Stochastic aspects: **No**

Modularity aspects: **High**

Supported Data Structures: **Complex**

Typical programming language types are supported (float, pointers)

Float Support: **Yes**

Model kind: **Graphic**

Flexibility

Backward 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: **Partial**

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

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

Modularity of the tool: **High**

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: **Mature**

Usability

Availability of Customer Support: **Yes**

Graphical User Interface: **Yes**

Mathematical Background: **Basic**

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

System properties are specified by graphically combining predefined operators.

No deep mathematical knowledge is 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 require a client account.

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

Company Constraints

Cost: **Pay**

Supported Platforms: **ALL**

Complexity of 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 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 into executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Textualimport**

NuSMV is essentially 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 Traceability: **No**

Project Management: **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 (Linear, Branching)**

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

Large-scale Verification Technique: **Bounded Model Checking, Symbolic Model Checking**

Model Based Testing: **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 non deterministic system initializations and transformations.

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.

Timing aspects: **Yes**

nuXmv supports the definition of timed transition systems.

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: **Complex**

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

Array variables are allowed.

nuXmv supports floating point numbers.

Float Support: **Yes**

Model kind: **Logical**

Flexibility

Backward Compatibility: **Likely**

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

Processes are deprecated and no longer supported in nuXmv.

Standard Input Format: **Open**

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

In the literature are found many cases of conversion from /to nuSMV

Modularity of the tool: **Low**

The framework supports various kind of model checkers,

Team Support: **No**

Maturity

Industrial Diffusion: **High**

many cases of industrial uses

Stage of Development: **Mature**

Usability

Availability of Customer Support: **Partial**

It is possible to 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**

Mathematical Background: **Medium**

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

Quality of Documentation: **Good**

Company Constraints

Cost: **Mix**

Usable only for non-commercial or academic purposes, need special agreement in case of commercial use

https://nuxmv.fbk.eu/index.php?n=Download.Download

Supported Platforms: **ALL**

Complexity of License Management: **Easy**

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

"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 into 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, which does support code generation.

Document / Report Generation: **Partial**

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

Requirements Traceability: **No**

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

Project Management: **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 (Linear, Branching), 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)

Large-scale Verification Technique: **SAT-SMT Constraint Solving and Theorem Proving**

Model Based Testing: **Yes**

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

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: **Internal, External**

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

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

Concurrency: **No**

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

Timing aspects: **No**

Stochastic aspects: **No**

Modularity aspects: **Low**

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

Supported Data Structures: **Complex**

Float Support: **No**

Model kind: **Imperative**

Flexibility

Backward Compatibility: **Likely**

Event B is a rather standard modelling language, even if ProB adopts its own syntactic 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: **Open**

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

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

Modularity of the tool: **High**

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

Team Support: **No**

Maturity

Industrial Diffusion: **High**

Stage of Development: **Mature**

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.

Mathematical Background: **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 Platforms: **ALL**

Complexity of 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 documenting 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:

https://www.atelierb.eu/en/
http://www.clearsy.com/en/our-tools/atelier-b/
https://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 into 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 doesn't 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 Traceability: **No**

Project Management: **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: **Theorem Proving**

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

Large-scale Verification Technique: **SAT-SMT Constraint Solving and Theorem Proving**

Model Based Testing: **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 allow internal nondeterministic behaviors.

Concurrency: **No**

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

Timing aspects: **No**

Stochastic aspects: **No**

Modularity of the Language: **Low**

Supported Data Structures: **Complex**

Float Support: **No**

Model kind: **Imperative**

Flexibility

Backward Compatibility: **Moderate**

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

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

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

Standard Input Format: **Open**

Event B specifications are encoded with a tool based flavour (Atelier B). Translators exist among the various Event B flavours.

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

Models can be exported to Rodin/ProB

Modularity of the tool: **Medium**

The tool is monolithic, but can be used in the abstract design, detailed 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 https://www.atelierb.eu/en/atelier-b-tools/).

Maturity

Industrial Diffusion: **High**

Stage of Development: **Mature**

Usability

Availability of Customer Support: **Yes**

provided by Clearsy for its maintenance edition

Graphical User Interface: **Partial**

Mathematical Background: **Advanced**

Property verification may require advanced theorem proving techniques.

Quality of Documentation: **Excellent**

Company Constraints

Cost: **Free**

https://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 https://www.atelierb.eu/wp-content/uploads/sites/3/atelierb/licenses/4.0/license-atelier-b-utilisation-en-V4.pdf)

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

Supported Platforms: **ALL**

Complexity of License Management: **Easy**

Zero cost license for the community edition (without 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 documenting 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 into 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 tool allows the visualization of sequence diagrams corresponding to selected interactive simulations or counterexample guided execution paths.

Requirements Traceability: **No**

Project Management: **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: **Graphical**

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

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

(http://people.cs.aau.dk/~adavid/ecdar/download.html)

Large-scale Verification Technique: **Statistical and Symbolic Model Checking**

Model Based Testing: **Yes**

(https://www.it.uu.se/research/group/darts/uppaal/download.shtml)

“Yggdrasil/Test case generator refactored and moved into Tools menu.“

Property Specification Language: **MITL**

Language Expressiveness

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

Name of Language: **Timed Automata**

Nondeterminism: **Internal/External**

Concurrency: **Sync**

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

Timing aspects: **Yes**

Stochastic aspects: **Yes**

Modularity of the Language: **Medium**

The system can be structured into a fixed set of processes.

Supported Data Structures: **Complex**

Float Support: **Yes**

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

Backward Compatibility: **Likely**

UPPAAL 4.x changes the language syntax but old versions of the models are still 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: **High**

Team Support: **No**

Maturity

Industrial Diffusion: **High**

Stage of Development: **Mature**

Usability

Availability of Customer Support: **Yes**

Provided by www.uppaal.com

Graphical User Interface: **Yes**

Mathematical Background: **Medium**

System properties are specified with a custom MITL temporal logics.

Quality of Documentation: **Good**

Company Constraints

Cost: **Mix**

Free for academic uses. Commercial licenses available.

Supported Platforms: **ALL**

Complexity of License Management: **Moderate**

Easy to Install: **Yes**

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Medium**

9 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 it is supported by the Computer Science Department of University of Oxford.

Web Sites:

https://cocotec.io/fdr/index.html

Documentation:

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

https://cocotec.io/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 into executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Textualimport**

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

Code Generation: **No**

Document / Report Generation: **Partial**

The tool 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 Traceability: **No**

Project Management: **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 possible evolutions of a process can be probed interactively (optionally Guided, interactive, Random)

Formal Verification: **Refinement Checking**

Large-scale Verification Technique: **Compositionality and Minimization, Partial Order Reduction**

https://cocotec.io/fdr/manual/cspm/definitions.html#csp-partial-order-reduction

Model Based Testing: **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: **Asynch**

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

Timing aspects: **Yes**

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

Stochastic aspects: **No**

Modularity of the Language: **High**

Supported Data Structures: **Complex**

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

Float Support: **No**

Model kind: **algebraic, functional**

Flexibility

Backward Compatibility: **Moderate**

The modelling language is rather stable and standard but the tool 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: **Open**

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: **Low**

Team Support: **No**

Maturity

Industrial Diffusion: **Medium**

Little evidence of industrial uses has been found.

Stage of Development: **Mature**

Usability

Availability of Customer Support: **Partial**

There is a mailing list for announcements and bug reports.

Graphical User Interface: **Limited**

Mathematical Background: **Advanced**

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 Platforms: **ALL**

Complexity of License Management: **Moderate**

Easy to Install: **Yes**

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Medium**

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

10 CPN-Tools

______________Information Part____________

Tool/Framework Name: CPN Tools

Description:

CPN Tools is an environment for editing, simulating, and analysing Colored Petri Nets. It was 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 was 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 into 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: **No**

Requirements Traceability: **No**

Project Management: **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 (Branching)**

Large-scale Verification Technique: **Bounded Model Checking**

Model Based Testing: **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: **Asynch**

Timing aspects: **Yes**

Stochastic aspects: **No**

Modularity of the Language: **High**

Supported Data Structures: **Complex**

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

Float Support: **No**

Model kind: **functional, graphic**

Flexibility

Backward 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 backward compatibility is however taken into due consideration by the developers.

Standard Input Format: **Partial**

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

Modularity of the tool: **Low**

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: **Mature**

Usability

Availability of Customer Support: **Partial**

Graphical User Interface: **Partial**

Mathematical Background: **Medium**

System properties are encoded as logical CTL formulae.

Quality of Documentation: **Good**

Company Constraints

Cost: **Free**

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

Supported Platforms: **Windows**

Complexity of 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**

11 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 date back to the mid 80s, since then 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 into executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Textualimport**

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 tool 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 Traceability: **No**

Project Management: **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 (Branching), Refinement Checking**

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.

Large-scale Verification Technique: **Compositionality and Minimization, Parallel Computing**

The supported size of "bcg" graph (explicitly modelling the evolutions of a system) is in the order of 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 approach to verifications allows the compositional analysis of large complex systems.

Model Based Testing: **Yes**

see JTorX, tgv

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: **Asynch**

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

Timing aspects: **No**

Stochastic aspects: **No**

Modularity of the Language: **High**

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

Supported Data Structures: **Complex**

Float Support: **No**

Model kind: **Imperative, algebraic**

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

Flexibility

Backward 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 arise (see http://cadp.inria.fr/changes.html). The framework provides conversion aids for specific incompatibility issues.

Standard Input Format: **Standard,** based on LOTOS

https://www.iso.org/standard/16258.html (LOTOS)
https://www.iso.org/standard/27680.html (E-LOTOS)

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

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

Modularity of the tool: **High**

The CADP framework is a collection of more than 50 programs performing various kinds of activities.

Team Support: **No**

Maturity

Industrial Diffusion: **Medium**

The tool is claimed to be used 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: **Mature**

Usability

Availability of Customer Support: **Partial**

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

Mathematical Background: **Advanced**

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 Platforms: **ALL**

Complexity of 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 dependencies 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.

12 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 Universiteit 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.htmlmucalc.html

https://www.mcrl2.org/web/user_manual/introduction.html

Documentation:

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

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 into 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 tool 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 Traceability: **No**

Project Management: **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 provided by LpsXsim

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

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

Large-scale Verification Technique: **Compositionality and Minimization**

Model Based Testing: **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: **Asynch**

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

Timing aspects: **Yes**

Uses positive real-time tags https://www.mcrl2.org/web/user_manual/language_reference/process.html

Stochastic aspects: **Yes**

(https://www.mcrl2.org/web/developer_manual/libraries/lts/classmcrl2_1_1lts_1_1probabilistic__state.html)

(https://link.springer.com/chapter/10.1007/978-3-030-17465-1_2)

Modularity of the Language: **High**

Supported Data Structures: **Complex**

Float Support: **No**

Model kind: **Algebraic**

Flexibility

Backward Compatibility: **Likely**

The tool is open source, and old versions 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: **Open**

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: **Medium**

The mCRL2 framework is a collection of more than 50 programs performing various kinds of activities.

Team Support: **No**

Maturity

Industrial Diffusion: **Medium**

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: **Mature**

Usability

Availability of Customer Support: **Partial**

Graphical User Interface: **Partial**

Mathematical Background: **Advanced**

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 Platforms: **ALL**

Complexity of License Management: **Easy**

Easy to Install: **Yes**

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**

13 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/
http://www.csl.sri.com/users/bruno/publis/sri-sdl-04-03.pdf
http://www.csl.sri.com/users/rushby/abstracts/sal-atg

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 into executable code, the production of accompanying documentation artefacts, and the SW development steps with which it interacts.

Specification/Modeling: **Textualimport**

SAL textual models are edited outside the SAL framework.

Code Generation: **No**

Document / Report Generation: **No**

Requirements Traceability: **No**

Project Management: **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 (Linear)**

Explicit 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)

Large-scale Verification Technique: **Parallel Computing, SAT-SMT Constraint Solving and Theorem Proving**

Model Based Testing: **Yes**

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 nondeterministic assignments.

Concurrency: **Asynch,Synch**

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

Timing 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 documentation of hybrid systems in HybridSal.

Stochastic aspects: **No**

Modularity aspects: **Medium**

Supported Data Structures: **Complex**

Float Support: **No**

Model kind: **logical**

Flexibility

Backward 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 Backward compatibility issues

Standard Input Format: **Open**

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: **Low**

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 industrial uses.

Stage of Development: **Mature**

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**

Mathematical Background: **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 Platforms: **ALL**

Complexity of License Management: **Easy**

Easy to Install: **Yes**

Note that a few missing (not found) dynamic libraries issues have to be fixed.

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Low**

14 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:

http://lamport.azurewebsites.net/tla/tla2-guide.pdf
https://lamport.azurewebsites.net/tla/toolbox.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 into 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 Traceability: **No**

Project Management: **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 (Linear)**

Large-scale Verification Technique: **Symbolic Model Checker, SAT-SMT Constraint Solving and Theorem Proving**

Model Based Testing: **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 non deterministic scheduling among processes.

Concurrency: **Asynch**

Processes are scheduled by interleaving.

Timing aspects: **No**

Stochastic aspects: **No**

Modularity aspects: **Medium**

Supported Data Structures: **Complex**

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

Float Support: **No**

Model kind: **logic**

Flexibility

Backward 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: **Open**

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

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

Modularity of the tool: **Low**

Team Support: **No**

Maturity

Industrial Diffusion: **Medium**

There are not many examples of industrial uses of TLA+, none of which railway related

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

Stage of Development: **Mature**

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**

Mathematical Background: **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 Platforms: **ALL**

Complexity of License Management: **Easy**

Easy to Install: **Yes**

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Low**

15 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 into 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 Traceability: **No**

Project Management: **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 (Branching)**

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

Large-scale Verification Technique: **On-the-fly Model Checking**

Model Based Testing: **No**

Property Specification Language: **Socl, 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 the rules defines deterministic 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 them can be nondeterministically selected for the system evolution.

Concurrency: **Asynch, 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 contain composite substates composed by parallel regions that evolve synchronously.

Timing aspects: **No**

Stochastic aspects: **No**

Modularity aspects: **High**

At the top level structure 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**

Heterogeneous, dynamically sized arrays are supported.

Float Support: **No**

Model kind: **Imperative**

Flexibility

Backward Compatibility: **Moderate**

Standard Input Format: **Standard** (based on UML State Machines)
https://www.iso.org/standard/32620.html

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

Modularity of the tool: **Medium**

Team Support: **No**

Maturity

Industrial Diffusion: **Low**

Stage of Development: **Prototype**

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**

Mathematical Background: **Medium**

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

Quality of Documentation: **Limited**

Company Constraints

Cos: **Free**

Supported Platforms: **ALL**

The desktop-based graphical user interface is supported only by macOS, but the tool can be used through any browser.

Complexity of License Management: **Easy**

Easy to Install: **Yes**

Railway Specific Criteria

CENELEC certification: **No**

Integration into the CENELEC process: **Medium**