Formal Methods in Safety-Critical Railway Systems

Conference Paper · August 2007

CITATIONS
64

READS
3,208

3 authors, including:

Thierry Lecomte
ClearSy System Engineering
54 PUBLICATIONS **418** CITATIONS

SEE PROFILE

Formal Methods in Safety-Critical Railway Systems

Thierry Lecomte [1], Thierry Servat [1], Guilhem Pouzancre [1]

[1] ClearSy, Aix en Provence, France.

{thierry.lecomte, Thierry.servat, guilhem.pouzancre}@clearsy.com

Abstract. *In this article we would like to present some recent applications of the B formal method to the development of safety critical systems, namely platform screen door controllers. These SIL3/SIL4[1] compliant systems have their functional specification based on a formal model. This model has been proved, guaranteeing a correct by construction behaviour of the system in absence of failure of its components. The constructive process used during system specification and design leads to a high quality system which has been qualified[2] by French authorities.*

1. Introduction

Historically, the B Method was introduced in the late 80s' to design correctly safe software (see [Abrial 96]). Promoted and supported by RATP[3], B and Atelier B, the tool implementing it, have been successfully applied to the industry of transportation. First real success was Meteor line 14 driverless metro in Paris: Over 110 000 lines of B models were written, generating 86 000 lines of Ada. No bugs were detected after the proofs, neither at the functional validation, at the integration validation, at on-site test, nor since the metro lines operate (October 1998). The safety-critical software is still in version 1.0 in year 2007, without any bug detected so far. Today, Alstom Transportation Systems and Siemens Transportation Systems (representing 80% of the worldwide metro market) are the two main actors in the development of B safety-critical software development. Both have a product based strategy and reuse as much as possible existing B models to develop future metros. For the time being, ClearSy has developed for Siemens the biggest B application: the Charles de Gaule airport shuttle automated pilot is a 150 000 lines of code program. On a different domain, Gemplus has developed a smartcard java bytecode verifier [Casset 99].

A more widely scope use of B appeared in the mid '90s, called *Event-B*, to analyze, study and specify, not only software, but also whole systems (see [Abrial 05]). *Event-B* has been influenced by the work done earlier on Action Systems by the Finnish School (Action System however remained an academic project). *Event-B* is the synthesis between B and Action System. Itt extends the usage of B to systems that might contain software but also hardware and pieces of equipment. In that respect, one of the outcome of *Event-B* is the proved definition of systems architectures and, more generally, the proved development of, so called, "system studies", which are performed before the specification and design of the software. This enlargement allows one to perform failure studies right from the beginning in a large system development. *Event-B* has been applied in many cases to various fields: certification of smartcard security policies (level EAL5+, Common Criteria),

---

[1] SIL (Safety Integrity Level) is defined as a relative level of risk-reduction provided by a safety function, or to specify a target level of risk reduction. Four SIL levels are defined, with SIL4 being the most dependable and SIL1 being the least. A SIL is determined based on a number of quantitative factors in combination with qualitative factors such as development process and safety life cycle management.

[2] French authorities define Qualified as « Certified and working » while Certified is mainly a verification of conformance to specification (the system produced may or may not work properly).

[3] Régie Autonome des Transports Parisiens : operates bus and metro public transport in Paris

verification of Ariane 5 launcher embedded flight software, generation of proven hardware specification, etc. *Event-B* has now its own modelling and proof platform Rodin[4].

In this article, we try to make clear what the different usages of B are, and to which extent its recent use in railway industry, at the system level, is a success. Section 2 presents the main differences between software development in general and software development in B. Section 3 details the first platform screen door system which has been modelled in B, details the development process and presents qualitative and quantitative results. Section 4 presents another PSD controller system under construction. Section 5 concludes.

2. Specifying safety-critical software or systems

Specifying safety-critical software or systems share the same philosophy: make use of abstraction, refinement and proof to mathematically demonstrate that a collection of models is coherent. First, internal coherency is checked for each model (described behaviour complies with properties). Second, each refinement is checked not to contradict its abstraction (the model containing fewer details). At the end, when this collection of models is proved, the concrete part is considered complying to the abstract specification and the model is then ready to be translated, as described in figure 1.

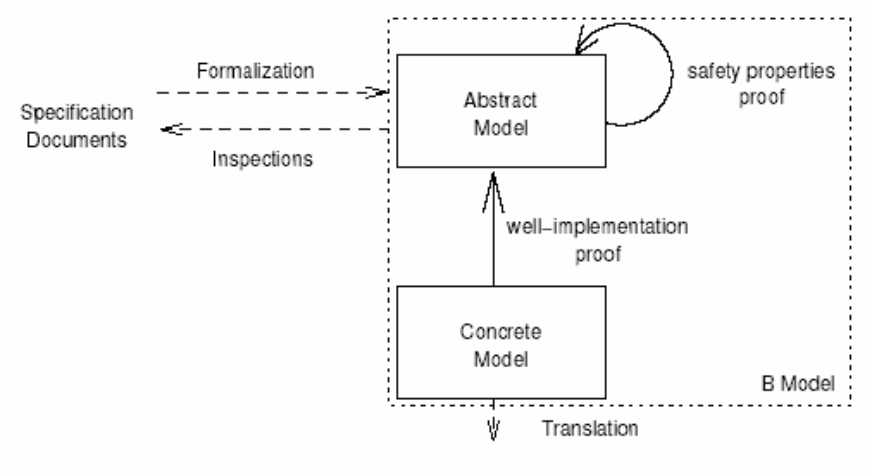

Figure 1. Modelling and proof cycle in B

Both approaches make use of set theory, first order logic and generalized substitution calculus. The main difference relies on the modelling paradigm and the way to structure the models. In software modelling, behaviour is described in terms of operations which represent programming functions that execute in sequence. The modelling language is different in specification and in implementation (no sequence in specification, no parallel action in implementation, no loop in specification, only implementable types in implementation, etc). Implementation B language is called B0. An implementation may import other models (abstract machines) and possibly delegate implementation of variables. That way, program specification is broken into smaller components that help to manage complexity. Design (refinement, decomposition with importation) are verified by proof on the fly, not when the development has been completed.

In system modelling, behaviour is described in term of atomic events that modify state variables of the system. One model represents a complete view of a closed system. The language is

[4] *http://sourceforge.net/projects/rodin-b_sharp/*

homogeneous during the complete development process; there is no specific language for final implementation. *Event-B* language slightly differs from B language, as it has been simplified and made unambiguous. This approach is well suited to represent asynchronous behaviour, such as interruption based software.

3. COPPILOT: A platform screen door controller

3.1. Presentation

In France, RATP has used for years platform screen doors (PSD) that prevent customers to enter or to fall on tracks. Such a system was adopted by the METEOR driverless metro, as it dramatically improves trains availability. In order to offer higher quality services and more safety to its customers, RATP was trying to introduce this kind of protection system in several lines, automated or not. For practical reasons, trains and cars could not be modified with the introduction of PSD. Before starting to deploy a new PSD system in an entire line, RATP initiated a project aimed at developing a prototype PSD system for three stations of line 13.

Figure 2. COPPILOT Platform Screen Door

These prototypes would be evaluated during eight months. ClearSy was in charge of developing the SIL3 compliant (probability of system failure less than 10^{-7}), control command controller. This controller is in charge of detecting the arrival, presence at a standstill and departure of trains without direct connection with them (on the contrary, Meteor[5] trains communicate with PSD through dedicated communications lines). Once the train is at standstill, the controller should be able to detect train doors opening and closing, and then issue PSD opening and closing orders. These orders have to be securely issued (failure by elaborating a wrong opening order may lead to customers injury or death), and controller have to be designed, tested and validated in accordance with railway regulations (IEC 50126, 50128, 50129 in particular).

The timescale of this project was quite short as the PSD controller would be installed in only 10 months after the beginning of the project. Given these strong timing constraints, we decided to adopt a secure architecture able to be quickly qualified by regulation authorities, loosely coupled

[5] First driverless metro operated in Paris

with sensor technology. The final architecture was based on Siemens safety automaton (Simatic S7), SIL3 compliant, and ordinary infra-red and radar sensors. In this case, security relies on the safety automaton and on sensor measure redundancy, not on the safety properties of the sensors. This approach leads to a decrease in system costs as usual since non-safety sensors are really cheaper than safety ones, leading to easier provisioning (shorter delay, no dependency towards a unique provider).

3.2. The development process

In order to reach the required safety level during project timescale, we decided to set up a development method aimed at reaching targeted reliability, and also ensuring traceability between the different stages of the projects in order to reduce the validation effort. This method was heavily based on the B formal method, and applied during most phases of the project.

Before any development activity, a formal functional analysis of the system was performed, to evaluate "completeness" and ambiguity freeness of the statement of work. At that time, the solution imagined by RATP was to point two laser telemeters on the platform, and to apply two independent 2D image recognitions in order to detect train arrival and departure, as well as train door opening and closing. The B method was used to:

- Verify on the overall system (PSD + controller) that functional constraints and safety properties were verified (no possibility to establish forbidden connections between train and platform or between train and tracks).
- Lead to the observation of dangerous system behaviour.

Telemeter based solution was then evaluated in order to verify that its compliancy with project constraints. This solution was finally abandoned due to the fact that designing two independent (but concordant) image recognition algorithms was judged too risky during the short lifetime of the project.

A new architecture was then imagined and proposed, making use of usual sensors and processing based on temporal sequence recognition of sensor events. Hyper frequency, infrared and laser sensors help to improve system resistance to various perturbations. Redundancy among sensors using different technology raises measures confidence. These sensors were positioned on the platform and pointed to the tracks in order to measure train position, train speed and train door movements.

System and software specification were then formalized in B by the development team, taking into account only nominal behaviour for the sensors (in absence of perturbation). Models obtained from previous functional analysis (independent from any PSD controller architecture) were directly reused. The proposed architecture was modelled and inserted in these previous models. New architecture was successfully checked by proof to comply with functional specification of the system, including parts of the French underground regulations. Controller functions were then precisely modelled (train arrival, train detection, train departure, train door opening, train door closing, etc). In the meantime, an independent safety case[6] was developed in parallel by the security team, in order to precisely define how external perturbations may influence the behaviour of the PSD controller. Perturbations were given a priori or a posteriori frequencies, depending on availability of such data at RATP, and a mathematical model, independent from the B model, was set up in order to determine quantitatively the security level of the system. A priori frequencies were

[6] Safety oriented study that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment.

verified during the eight month experiment. In case these frequencies were not verified and lower system security below SIL3 level, the PSD controllers would have to be redesigned considering this new information.

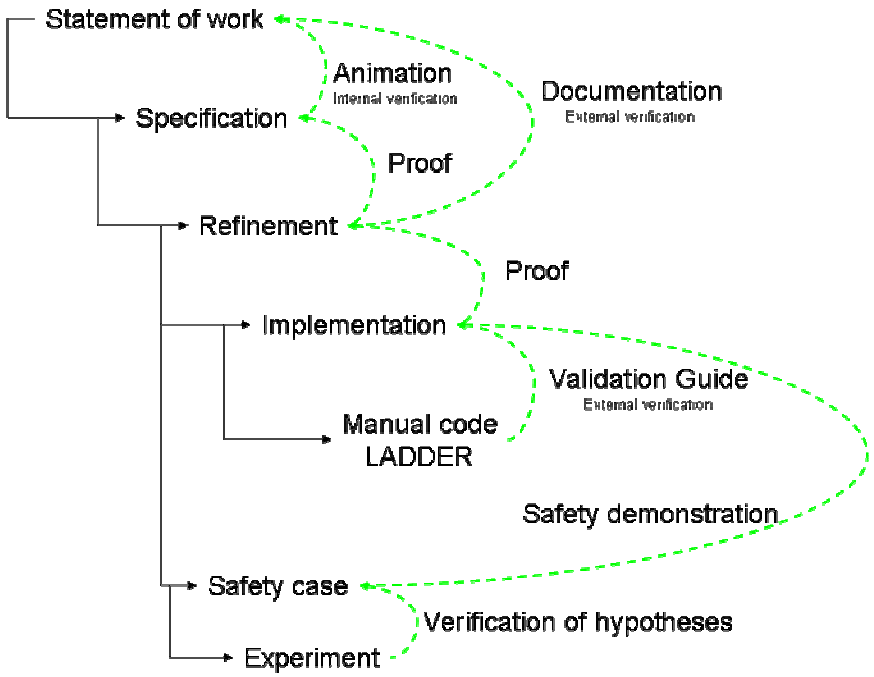

Figure 3. Development and verification process

The resulting B model was animated with the Brama animator[7], in order to verify on given scenarios that the model produced was corresponding to the real system we were modelling. This model animator was not part of the validation process, as this would require it to be qualified as a SIL3 software, but it helped us to check models against reality and to internally verify their suitability. In figure 4, the left window displays the list of events of the model (fireable ones have a checkbox before their name) and the list of variables and associated values. Values can be modified on the fly to setup a starting point, the animator verifying that proposed values comply with the invariant properties.

Specification documentation was partly elaborated from the system level models developed during this project. The composys[8] tool has no proof capabilities but, as an engineering tool, helps the modeller to add contextual information (comments, description, component name, etc) in B models that are used to generate in natural language the specification documentation describing the complete system. As events are associated to components and as variables are used within events (read/write), Composys computes relationships among components constituting the system being modelled, depending on how variables are read or modified.

[7] http://www.brama.fr

[8] http://www.composys.fr

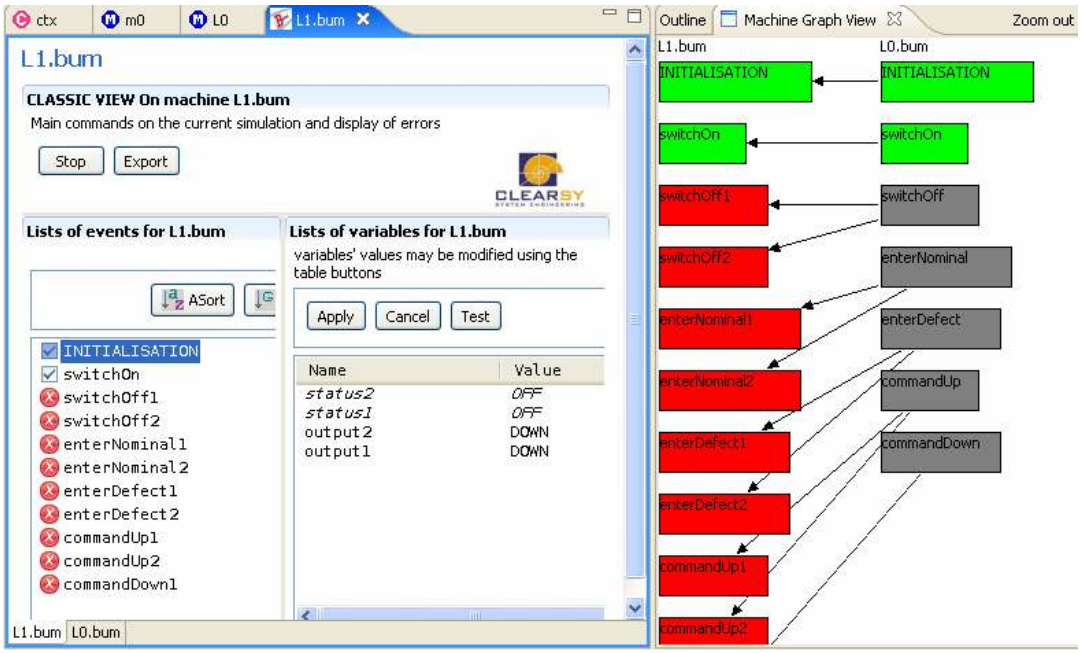

Figure 4. Animation of B refinements with Brama

These relationships are then graphically displayed, boxes represent components, ovals represent medium linking two or more components (medium can be an electric wire, a network, etc) and text near connecting lines are description of variables involved in the information exchange (Figure 5). This document was used to check models with experts of the domain, unable to read and understand formal models.

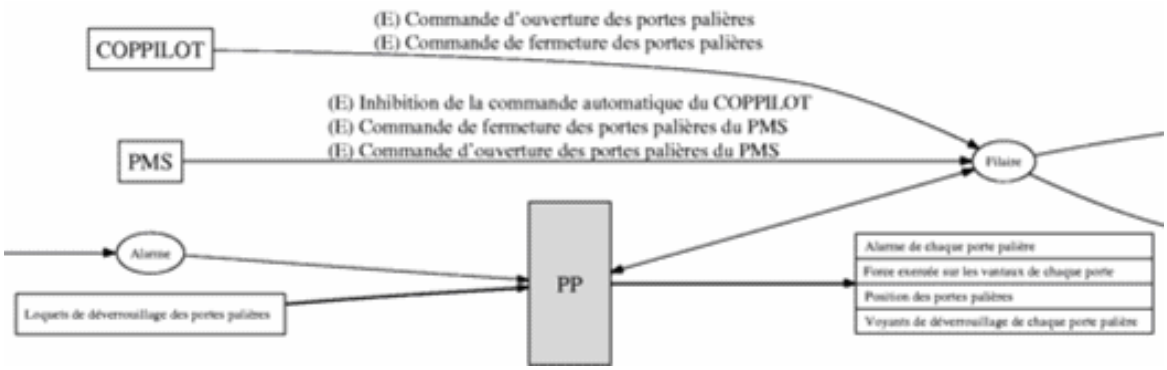

Figure 5. Example of relationships within a B model exhibited by Composys

The development of the software was based on the formal models, as B enables the production of source code, proven to comply with its specification. Siemens automaton can be programmed in the LADDER language but, unfortunately, requires entering program source code via its graphical interface (according to its certificate) to keep its SIL3 accreditation. A dedicated translation schema (from B to LADDER) was elaborated. B to LADDER state diagrams translation is straightforward and some optimisations were introduced in order to verify temporal constraints (cycle time in particular). During validation phase, one can determine which event of the B model corresponds to the path of the LADDER program for a cycle (a LADDER program is defined by logical equations and is analyzed in term of execution path). In case the source code is automatically generated by a

qualified translator (as for automatic pilots, by Siemens and Alstom), no unit test is required, this testing phase being covered by the proof of the model. In this project, as the source code was not generated automatically by such a translator, test was required and test specification was elaborated by usual means. Some months after the beginning of the project, we obtained a fully functional, tested and validated application. The process described above has enabled us to produce a 100% tested, error free (against its specification) software when running validation test bench for the first time. A dedicated test bench was designed to simulate major perturbations (sensors were emulated) and run during days, but no faulty behaviour was observed.

Integration testing was performed on a dedicated testing platform installed in the METEOR line. Tracks and sensors being already protected by PSD in the line, measurement campaigns were setup quickly in order to assess as quickly as possible security, availability, response time, etc). Sensor technology choices were validated at that occasion.

3.3. Results

Finally, 4 months after the beginning of the project, the PSD controller was deployed on 3 platforms on line 13, for a 8 month experiment. The following metrics were obtained:

- equip: a project manager, a developer, a validation engineer, a safety engineer
- initial system level functional specification: 130 page document
- safety case: 15 documents representing 300 pages
- development documentation: 30 documents representing 600 pages
- formal B models: 3500 lines of code, representing about 1000 proof obligations. 90 % were demonstrated automatically, the remaining proof obligations were discharged in two days with the Atelier B interactive prover.

This system was experimented during 8 months, controlling around 96 000 trains. No fault was observed. Hypotheses made during the safety case were confirmed and made more accurate. System availability conformed to expectations and, after initial setup and tuning, no passenger remained stuck in the train (PSD should open 9999 out of 10000 times when a valid train is at standstill opening its doors).

4. DOF1: another platform screen door controller

At the end of the experiment, ClearSy was given the responsibility to design a similar system: the DOF1 PSD controller system. In the context of the "Paris Subway Line 1 Automation" project, the SIL4 DOF1 safety system, which is independent from the automatic train system, will command the opening and closing of the platform screen doors installed on all the line's platforms.

This system will be used with existing trains and will be compatible with the new automatic trains that will progressively replace the current ones. DOF1 also prevents the train doors on the opposite side of the platform from opening. DOF1 will be installed on 26 stations and 52 trains.

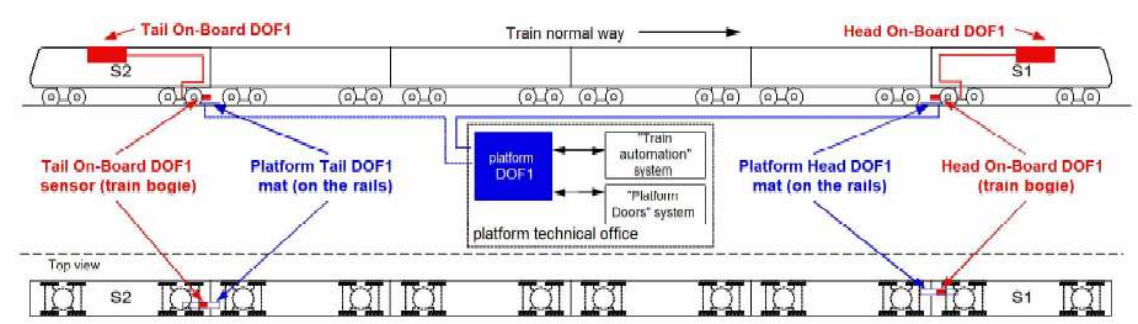

Figure 6. The DOF1 architecture

DOF1 will then be disassembled when all the automatic trains are in operation on Line 1.

The DOF1 system includes an embedded portion on the train that processes the train door opening and closing commands issued by the conductor and sends the command to the portion located in the platform's technical office. The command to open the door is processed and sent to the landing doors. This architecture is presented in figure 6.

This system is SIL4-level safe to ensure the landing doors can only be opened when the train has reached the platform. The solution is based on Siemens SIL3 automatons.

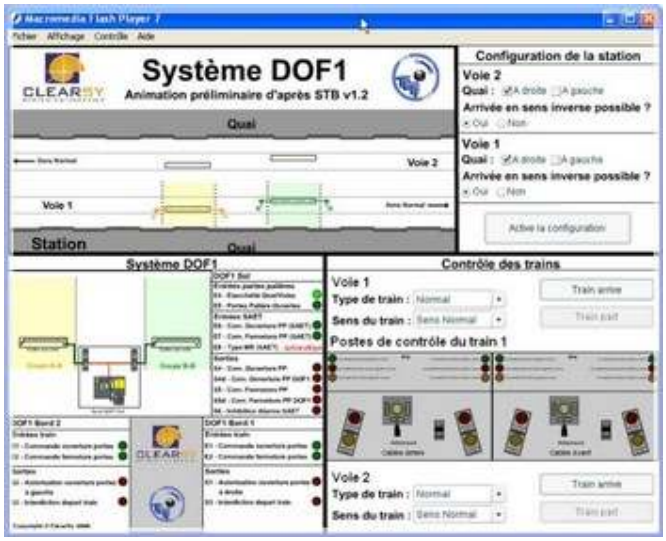

Figure 7. The DOF1 animated model submitted to the call for tender

We adopted an original method to submit to the call to tender: a B model of the specifications was constructed with the Composys tool, then graphically animated with the Brama tool (figure 7). We were therefore able to define the needs of the RATP by transcribing our understanding of the system into a model and then validating this understanding by animating the system in various scenarios viewed on the screen. Questions could therefore be asked and a detailed response provided, as the system must be designed in only six months.

As for the COPPILOT system, we use a development process that integrates the B method, from the system specifications up to the code stage. The B models developed participate in demonstrating the system's safety and the level of availability of the system that must be very high in order to ensure fluid traffic.

5. Conclusion

The methodology we have developed appears to be efficient and well suited to address projects requiring high level of safety and short development time. The B formal method was not initially considered by RATP, but is now well accepted. The writing of some extra documents were required to help RATP engineers to fully understand, verify and qualify our deliverables. Reuse of existing models for similar projects proved to be efficient.

Several issues have to be overcome in order to obtain a more efficient process:

- Manual translation from B models to the target language (LADDER in our case) is error prone and require specific verification process (a dedicated verification guide was written, enabling RATP to cross-check software). One possible solution is to develop a dedicated translator (from B to automaton assembly). That translator should be certified. Another possibility is to use a model animator (like brama) to automatically produce test benches and to validate manually developed applications on the final target.
- Functional and dysfunctional models are disjoint and also error prone. Research project related to the combination of B and Altarica [Rauzy 98] (used by EADS on A350 Airbus program for reliability studies) languages is ongoing.

Acknowledgements

The authors would like to acknowledge the support of the RODIN (Rigorous Open Development Environment for Complex Systems) Project funded by the European IST.

References

Abrial, J.R. (1996) , *The B-book: Assigning programs to meanings,* Cambridge University Press

Abrial*,* J.R (2005)*., Rigorous Open Development Environment for Complex Systems: event B language*

Burdy, L.(1996) *, Automatic Refinement. In Proceedings of BUGM* at FM'99

Casset, L.*,(1999) A formal specification of the Java byte code verifier using the B method,* Lisbonne 99

Rauzy*, A., (1998),The Altarica Language* In Proceedings of the International Conference on Safety and Reliability, ESREL'98,

Sabatier, D. & al (2006), *Use of the Formal B Method for a SIL3 System Landing Door Commands for line 13 of the Paris subway*, Lambda Mu 15