Computers in Railways VII, C.A. Brebbia J.Allan, R.J. Hill, G. Sciutto & S. Sone (Editors)

Validation and verification of METEOR safety software

J.L. Boulanger & M. Gallardo
ESE/ICH/AQLM, RATP, France.

Abstract

The study described in this paper was conducted by the Paris Public Transport Authority (RATP) and aimed to verify the correct application of the concept by the establishment of lists of tests to be played on the final software. In this article, we present the validation process which has been chosen by the RATP team in charge of fixed equipment for the automatic Paris metro system *METEOR* (Métro Est Ouest Rapide – high-speed east-west metro). The idea of properties which the software must respect is one of the characteristics of this validation process. This article gives an idea of industrial practices within a context of installing a safety-critical system implementing formal methods such as ASA+ and the B method. *METEOR*'s complexity will be quantified throughout the article, using information such as the number of B components, the number of lines of ADA code produced by the code generator and the number of tests performed by the RATP.

1 Introduction

The *METEOR* line is managed by the SAET (Système d'Automatisation de l'Exploitation des Trains – Automatic train management system) which is a complex distributed real time system, the main function of which is to provide passenger transport, while guaranteeing a very high level of safety for passengers. We must therefore be certain that it meets safety operating criteria ([6]). The real time nature of the system also reflects the fact that it interacts with a physical environment, the behaviour of which is, by definition, *uninterruptible* and *irreversible*. It must therefore also meet strict temporal constraints and be able to react *quickly enough* to any event caused by the environment.

Computers in Railways VII, C.A. Brebbia J.Allan, R.J. Hill, G. Sciutto & S. Sone (Editors)

The safety functions on the new automatic Paris metro line, the *METEOR,* are provided by digital processes, which require very strict operational safety criteria (about 10^{-9}). Matra Transport International (MTI) is in charge of development on behalf of the RATP. RATP's validation policy is based on the implementation of a double validation process using different methods from those applied by the constructor. In this article, we shall present the approach used as well as the resources installed, with the aim of validating the safety functions of each item of safety equipment. This validation concerns functional design as well as final product verification. As with the validation of MAGGALY ([13]), the validation process installed by RATP is based on the idea of properties ([16] and [18]).
This article is organised in three parts: in section 2, we give a brief description of the principles used for the *METEOR* line, section 3 describes the development process used by Matra Transport International and the validation process which has been installed at RATP is described in section 4.

2 Presentation of METEOR

The new *METEOR* line will be used for both trains not equipped, which require a driver and trains equipped which can be controlled automatically and run with or without a driver. The equipped trains have a choice of two modes, manual drive (referred to as MD) and integral automatic drive (referred to as IAD). The line is managed by an automatic control system fitted to the length of the line and in the equipped trains. The line (see Figure 1) is managed by an *Automatic Line Controller* (ALC) and is split into automated sections ([12]), each section being managed by a computer, called an *Automatic Section Controller* (ASC). Equipped trains have an onboard computer called the *Automatic Onboard Controller* (AOC) which, amongst other things, regularly sends a location signal to the ground automatic controller in charge of operations. Computer processing is cyclic and uninterruptible.

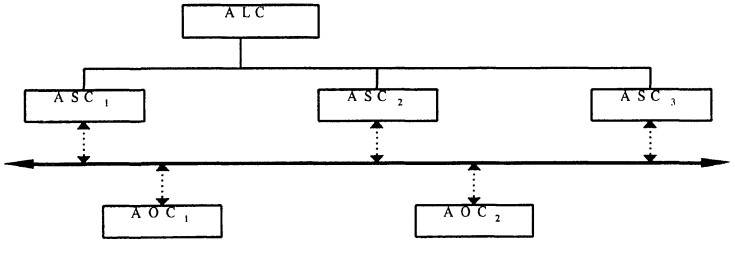

Figure 1. Automation system architecture.

The main function of the automatic control system is to give movement orders to all kinds of trains. For manually driven trains, the orders are transmitted through the signalling system. For automatically controlled trains, the orders are transmitted in the form of messages defining the target to be reached. A target is a physical point on the line which the train in question must not go past. Target

generation requires complex safety-dependent calculations and is performed by the *anticollision* function which is designed to prevent collisions between trains, whether they are in automatic or manual drive. The control system has full control of trains in automatic mode (acceleration, emergency stop, etc.), but manually driven trains are autonomous and can override an order to stop transmitted by the signals. The control system is therefore split between the various parts of the equipment. The *Automatic Section Controller* is in charge of the *anticollision* process and transmits targets to the *Automatic Onboard Controllers*, which they must respect. *Automatic Onboard Controllers* also perform the important job of transmitting a precise location as often and as regularly as possible.

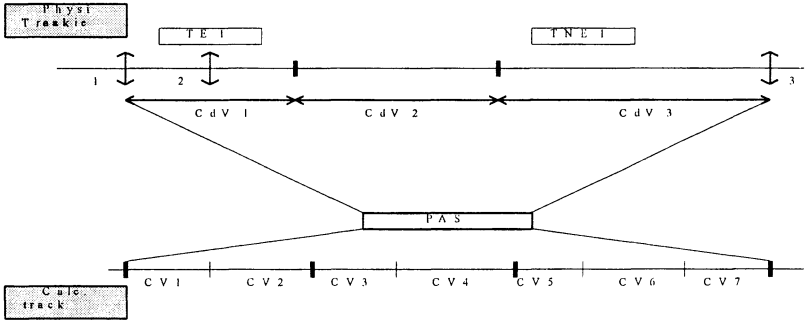

Figure 2. An automated section.

The anticollision system is essentially based on the idea of fixed blocks ([7]), routinely used in the rail sector. A block is a section of railway track, which can only be occupied by a single train at any given time. The presence of trains is detected by sensors called *Track Circuits (TC)*. Common usage has led to the section of track being controlled also being called the *Track Circuit*. The section of track thus controlled has a minimum length determined on the basis of technical and economic constraints, which consequently limits railway capacity. To overcome this disadvantage, the idea of *Virtual Block (VB)* is used as well as *Negative Detector* (optical barriers referred to as ND) which provide more precise train location. The virtual blocks are a subdivision of the *Track Circuits*. A *Virtual Block* is a section of track on which the presence of trains is not only determined by sensor status (*Track Circuits, Negative Detector*) but also by distributed dynamic calculation, performed by the control system according to train speed and track characteristics.

Virtual block status is calculated during each cycle and used to obtain a map of the track in terms of actual or potential occupation of virtual blocks by trains equipped or not, according to the status of the *Track Circuits*, *Negative Detector*, location messages received and *Virtual Block* status during the previous cycle. The example shown in figure 2 defines a very simple automated section, physically consisting of 3 *Virtual Block*s with 3 *Negatives Detector*s represented by double vertical arrows. *Negatives Detector*s 1 and 3 represent the respective

Computers in Railways VII, C.A. Brebbia J.Allan, R.J. Hill, G. Sciutto & S. Sone (Editors)

input and output for the section managed by this *Automatic Section Controller*. The *Virtual Blocks* represent the *Automatic Section Controller* 's interpretation of train occupation of the track.

3 Presentation of the development process

3.1 Process

The entire METEOR development system is presented in [6]. Bear in mind that the study outlined in this paper concerns the validation of safety-critical software for one device (*ASC, ALC or AOC*). The development process presented therefore applies to the development of this type of software.

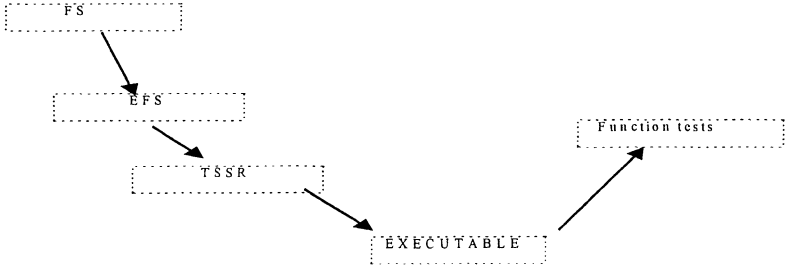

Figure 3. Development Process.

The process starts with consideration of a functional specification (FS) which is then broken down into functional specifications for each device. From this point, software design can begin with the introduction of a technical specification of software requirements (TSSR) which will then be modelled using B method ([1]).

From the description of a line (figure 2 for example), a set of data can be deduced which will be common to all the equipment in the system. These data are based on line topology and train characteristics and are called the *Topological Invariants* ([9]). From the description of software requirements (TSSR) for a device, a B model can be built and a set of data called the *Requirement Invariants* defined, this being derived from the *Topological Invariants*. *Requirement Invariants* are associated with a specific device. The B model is automatically translated into secured ADA language by the tool. Separating data and code (figure 4) is used, first to eliminate any compatibility errors between devices, and, above all, to provide a generic description of the equipment.

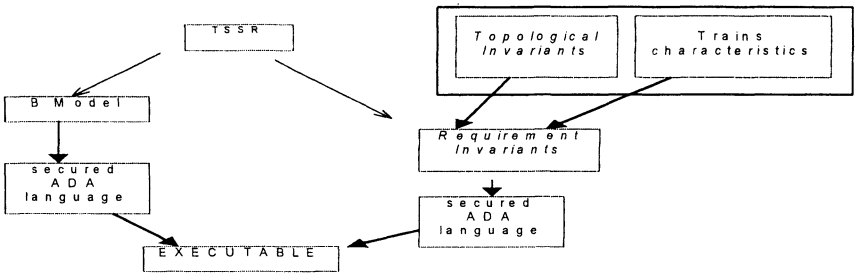

Figure 4. Separation of code and data.

The MTI development process shown in figure 3, includes the use of the formal B method. You can see that all the validation phases (Unit Tests, Integration Tests and Function Tests) have been reduced to simple function tests.

3.2 Using B method

In the French rail sector, the use of formal methods, notably B method, is increasingly common fore the development of safety-critical systems. Software used for these safety-critical systems (rail signalling, automatic drive), must meet very strict criteria with respect to quality, reliability and strength. One of the first applications of formal methods was on the SACEM ([10]) a posteriori. More recent projects such as the CTDC, KVS or METEOR ([2] et [4]) which is in the process of being extended, use the B method (figure 5.) throughout their development (from specifications to code).

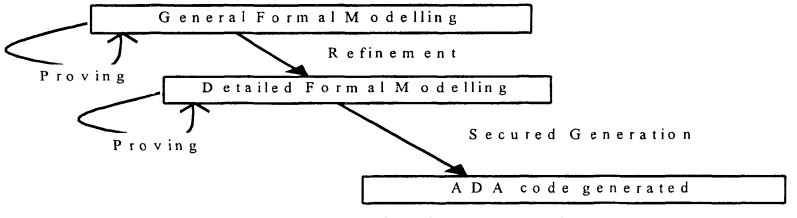

Figure 5. B development cycle.

The B method was developed by Jean-Raymond Abrial ([1]). It is a formal model-oriented method like Z or VDM, but allows incremental development from specification to code, using refinement [14], in a single formalism, abstract machine language. At each stage of B development, proving obligations are generated to guarantee the validity of the refinement and consistency of the abstract machine. MTI and the RATP use "Atelier B" developed by Stéria Méditerranée. The software is produced in the form of a layered model ([3]) which consists of a general formal model and a detailed formal model, both made up of different levels of abstract machines (machine, refinement and implementation). The validation phase consists of generating the set of obligations to prove coherence and refinement and successfully proving this set

of proof obligations. The last level of refinement is called "installation" and is described in a subset of B called B0 which can be directly translated into a conventional language such as ADA or C. The paradigm for modelling in B is based on the introduction of properties which the model must verify, and which are then refined. METEOR's automatic control system, split between the three devices (AOC, ALC and ASC), includes 1150 B components, i.e. approx. 115 000 lines of B code which generated 27 800 proof obligations. All these proof obligations have been proved. The application code (generated from B code) and data give a total of 150 000 lines of ADA code.

4 Validation Process

4.1 Presentation

RATP's policy in the safety software area is to perform a double validation independent of that used by the suppliers, leading to the process described in figure 6. RATP validation is based on function testing. There are two methods for choosing test sets, using either selected or generated test input. In our case, the models relate to a function so we use selected functional test input.
Equipment is validated in two phases as is the case for development. First of all the safety-critical software for a device is validated by function tests and then all the safety data for this item of equipment is validated.

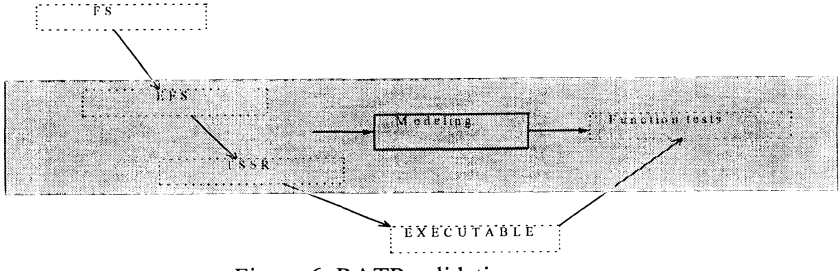

Figure 6. RATP validation process.

The function tests are performed using targets; this means that each test is run on a tooled version of the equipment to be tested, which is called a "Test Bench". The test bench consists of a set of hardware and software including line equipment (AOC, ALC or ASC) and an environment simulator.

4.2 Safety-critical software validation process

4.2.1 Properties

The safety-critical properties are formulated using properties of right (liveness) and wrong functions (safety). The right function properties are the global properties applied to all the system's inputs/outputs: there are properties for comparing the physical (environment status) and calculations (status of virtual

blocks, buffers and targets), properties concerning persistence from one cycle to the next or not, and temporal tree-structured properties which are transversal to the various *Anticollision* functions. Wrong function properties are listed in the form of situations which must not arise (following incoherence of virtual block statuses, target overtaking another train...).

4.2.2 Process

The method used consists of drawing up a set of formal properties (based on mathematical logic), and inserting them into a model of the system or sub-system to be validated. This model must be installed in a computer tool used to reason and run test script simulations. The set of formal properties is used to produce an accurate, unambiguous and demonstrable description of the behaviour of the software being validated.

The validation process implemented by the RATP works as follows:

1. Analysis of specification documents (EFS and TSSR) to:
   a) produce a critical analysis of each function
   b) produce a list of safety function properties.
1. Model safety-critical functions (in association with the TSSR);
2. Verify the properties by implementing them in the model or verifying them on the model.
3. Produce test sets.
4. Run test sets on test benches.
5. Use ASA+ models such as Test Oracle.

All these properties are formalised in a validation document. In [5], we presented the entire validation process and its results for the *Anticollision* function, which is one of the safety functions on the Automatic Line Controller equipment.

4.3 Principle for modelling safety functions

Using a program P, its specification PS and a set A of safety properties in P, expressed as variables of P, a model M can be designed, with input E and Boolean output S and S'. Formal verification of a program, P then consists of simulating (interactively or exhaustively) the model using input to verify that the right output is always obtained (figure 7.).

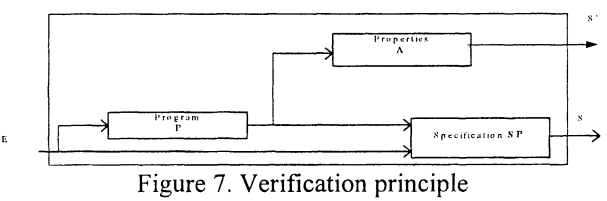

Figure 7. Verification principle

To facilitate the verification phase, it is preferable to close the model by introducing an environment which will let you generate input and will react to the output. The dynamic behaviour of this type of model in its environment is shown in figure 8.

Computers in Railways VII, C.A. Brebbia J.Allan, R.J. Hill, G. Sciutto & S. Sone (Editors)

Since the validation was performed for a given item, and in view of the size of the equipment ([4]), it was decided not to produce one global equipment model but one model per function. The link between functions was provided through traceability at interface level. The validation models were produced using the VERILOG ASA+ method and an ELSIR Petri net modelling tool. In this presentation we are interested in the ASA+ modelling tool.

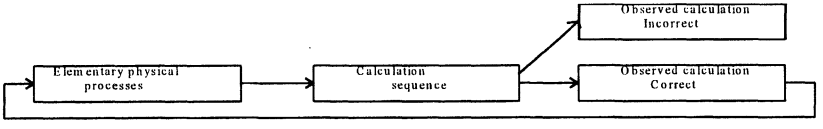

Figure 8. Execution principle.

4.4 Presentation of ASA+

The ASA+ method ([19]) is based on a language used to specify and design a system, and a modelling and simulation environment. ASA+ is based on ASA (Analysis Structured by Automaton), the ISO standard language ESTELLE ([8] and [11]) and its extension ESTELLE*. ASA and ASA+ are used for railway applications as well as in the electrical sector, see [15] for an account of experiments in this field. The ASA+ method is based on SADT type functional analysis and behaviour modelling in the form of extended state transition systems. The functional analysis is the static part of the model; it is structured through modules, interface points and communication channels between the interface points. Each module represents a type of behaviour which can be broken down into other lower level types of behaviour. The process is therefore repeated until the basic behaviour represented by the sheet modules is obtained. Modules communicate via communication channels linked by interface points. The modules communicate with each other by exchanging messages, transmitted or received by the respective interface points and propagated through the channel linking the modules. Communication can be synchronous (rendezvous) or asynchronous (file). The static view (figure 9.) provides two points of view: the hierarchical point of view which is used to display the breakdown structure, and the composition point of view which is used to show communications between modules at the same level; this promotes the black box module concept.

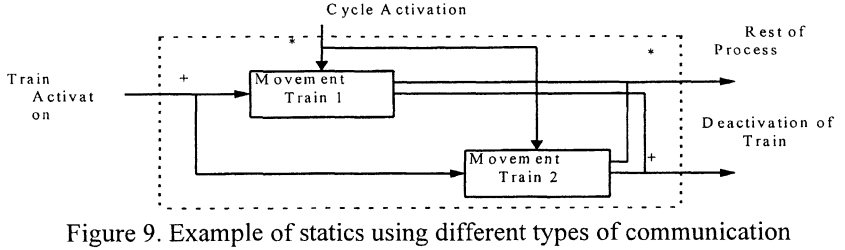

Figure 9. Example of statics using different types of communication

With synchronous communications, you can choose two methods of rendezvous, rendezvous with all (all recipients are waiting), symbolized as * or rendezvous with at least one (at least one recipient is waiting) symbolized as +.

4.5 Using models for verification

Since each model relates to a function, we produced a set of function tests by selecting test input. These function tests are then translated and run on the target. The target is in the form of an environment including a test bench and a simulator.
RATP validation begins with modelling (ASA+) each function, so that the principles can be validated. The test book for each function is produced on the basis of acquired experience and the model ([5]). Each example of a function test is then encoded and simulated on the equipment test bench.

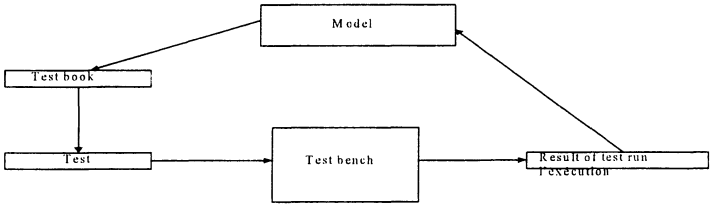

Figure 10. Verification and validation chain.

Validation (figure 10.) consists of examining the results of running each test script, then deciding whether or not this result is correct. The difficulty is in choosing a baseline on which a decision for correction can be based. A test oracle is obviously required. In our case we use the ASA+ model.

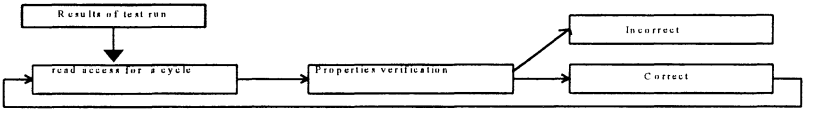

Figure 11. Verification principle.

To verify the results of a test, the environment model and function were replaced in the ASA+ model by an automatic device which was interfaced with code providing read access for a cycle in the result file for the test script being validated.

4.6 Data validation

There are three types of METEOR equipment (AOC, ALC and ASC) and the line is split up into several ASCs. To avoid any ambiguity between devices, it was decided to use a single database. The data ([7]) describe track topology (*Track Circuit,* Virtual block, *Negative Detector,* points, etc.), train

Computers in Railways VII, C.A. Brebbia J.Allan, R.J. Hill, G. Sciutto & S. Sone (Editors)

characteristics (length, kinematic information, etc.) and system constants (e.g. cycle times).

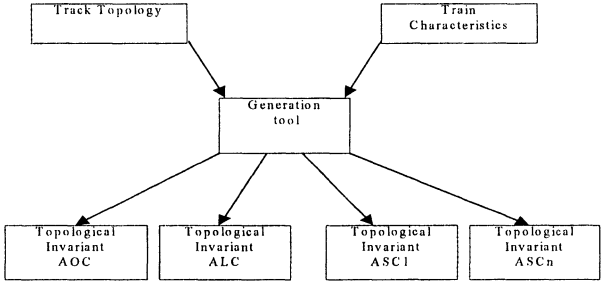

Figure 12. Generation sequence.

The safety-critical software for a given device, generated from the B model, uses the equipment's topological invariants. The B model was proved by introducing statements to characterize these topological invariants. These statements are processed as specifications of the constructor's generation tool (figure 12.). This process lets you separate code from data which allows you to have a purely generic code. The RATP validation process is based on the idea that the generation process can be reversed, thus demonstrating the exactitude of the data by comparison (figure 13.). Validation by a tool which, using the equipment's topological invariants, generates primary data by reverse transfer, is completed by verification of safety-critical properties, taken from either specifications or the reversal process. Reversing the generation process alone has been used to identify a great many errors. The data validation tool has been used to identify real problems and also track configurations which are acceptable although not planned.

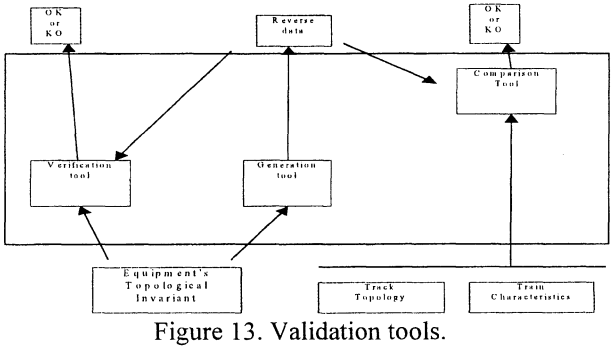

Figure 13. Validation tools.

Data-linked safety-critical properties are input into the verification tool using a formal set-theoretic language. Constraints are verified by exploration (case by case proving).

Computers in Railways VII, C.A. Brebbia J.Allan, R.J. Hill, G. Sciutto & S. Sone (Editors)

5 Report

To validate the safety-critical software for the *METEOR* Automatic Control System, the AQLM produced:

- 20 Principle files;
- 23 Models;
- 30 Test books;
- More than 5 000 tests in a simulated real time test environment.

During the development cycle and in parallel with the constructor's validation processes, it revealed:

- 400 criticisms of safety at specifications level;
- 110 anomalies in the safety-critical software as a whole (3 versions in 3 different applications).

Of course, all safety-critical anomalies were corrected before the latest version of the software was released.

6 Conclusions

This study allowed us to demonstrate the feasibility and efficiency of a validation process based on the idea of safety-critical properties. The validation process based on properties is an external verification process which uses static and dynamic verification. Indeed, static verification, with no actual activation, was performed on a behavioural model of the system (Finished extended state transition automaton) which provided a behavioural analysis. Dynamic verification involved activating the system by providing evaluated input and was performed using a set of function tests for equipment on a test bench. The validation process proposed was similarly applied to the safety data for each device.

References

[1] J-R Abrial, "The B Book – Assigning programs to meanings", Cambridge University Press, ISBN 0-521-49619-5, 1996

[2] P. BEHM, "Application d'une méthode formelle aux logiciels sécuritaires ferroviaires", ATELIERS LOGICIEL TEMPS REEL, 17-19 November 1993.

[3] P. BEHM, "Formal development of safety critical software of METEOR", First B Conference, Nantes, France, November 24, 25, 26 , 1996.

[4] P. BEHM, P. Desforges et F. Meija, "Application de la méthode B dans l'industrie ferroviaire", ARAGO 20, page 59-88.

[5] JL. Boulanger, V. Delebarre, S. Natkin, "Validation de Spécification basée sur un modèle formel: Application à un système de transport ferroviaire", Rapport CEDRIC No 97-17, December 1997.

Computers in Railways VII, C.A. Brebbia J.Allan, R.J. Hill, G. Sciutto & S. Sone (Editors)

[6] A-M.Chaumette and L. Le Fevre, "Système d'Automatisation de l'exploitation des trains de la ligne METEOR", REE No 8, September 1996.

[7] G. Conan, M. Devillechabrolle, "La logique de canton de METEOR", Revue générale des chemins de fer, pages 35-36, No 6 - June 1996.

[8] Courtiat JP, Deminbinski P et Groz R, "Estelle, un langage pour les algorithmes distribués et les protocoles", TSI May 1987.

[9] J-P Georges, "Principes et fonctionnement du Système d'Aide à la Conduite, à l'Exploitation et à la Maintenance(SACEM). Application à la ligne A du RER." Revue générale des chemins de fer, No6, June 1990, DUNOD.

[10] G. Guihot, C. Hennebert, "SACEM software validation", in Proc. 12th IEEE-ACM International Conference on Software Engineering: March 1990.

[11] ISO, *Estelle, A Formal description Technique based on an Extended State Transition Model* ISO IS 9074, 1989.

[12] P. Lecompte, PJ. Beaurent, "Le système d'automatisation de l'exploitation des trains (SAET) de METEOR", Revue générale des chemins de fer, pages 31-34, No 6 - June 1996.

[13] A. Maire, "Présentation du système MAGGALY", Symposium international sur l'innovation technologique dans les transports guidés, ITIG'93, Lille, September 1993.

[14] C. Morgan, "Deriving programs from specifications", Prentice Hall International, 1990.

[15] J-P Sabathé, "Premier retour d'expérience sur le noyau de vérification formelle de l'outil ASA+", Génie Logiciel No 45 September 1997.

[16] A. Stuparu, F. Baranowski et P. Ozello, "INRETS experience of the highly critical software validation of the MAGGALY subway", IEEE – ISSRE'93, Colorado, November 1993.

[17] A. Stuparu, F. Baranowski et P. Ozello, "La validation fonctionnelle des logiciels de sécurité du métro automatique maggaly", Recherche Transports Sécurité, No 42, March 1994.

[18] Stuparu A, " Maggaly , Métro Automatique de Lyon, validation complémentaire des logiciels de sécurité ", Journée d'électronique (JE 95) 1995.

[19] VERILOG, "Langage LSA+- Manuel de référence", December 1994.