Practical formal validation method for interlocking systems

M. Antoni a – N. Ammad b

a SNCF – Infrastructure Direction – Maintenance Engineering – Technological Innovation Department – 18 rue de Dunkerque 75018 Paris – France – e-mail : marc.antoni@sncf.fr - Phone (+33) 6 29 91 77 43

b SNCF – Infrastructure Direction – RAMS division of Signalling department – 6 Avenue François Mitterrand 93574 La Plaine Saint Denis Cedex – France – e-mail : nadia.ammad@snclf.fr - Phone (+33) 1 55 93 30 14

Abstract

Today, the main issue is to answer the following question: have we finally recognized that when it comes to software, the delivery of absolute numerical safety targets is considered to be impossible, and the methods contained in the CENELEC standard produce a “probability” that certain unsafe failure rates will be archived, rather than an absolute assurance? We know that checks that are undertaken (and their results) before putting safety signalling facilities into service are essential but they are time consuming. And there is no guarantee that these tests are exhaustive, particularly for computerised equipments. In the context of greater economic constraints and increasing complexity of computerized tools, the capacities of the classic approval process are today attained. In practice there is a reduction in the validation cover rate which results in more and more numerous unsafe failures. This paper assumes that it is possible in practice to give an exhaustive formal proof that the “functional” of the signalling application (functional “white box”) is safe in the context of use (over-system). The presented method makes it possible, after a rigorous and cost effective design, to formally validate the “functional” software of critical computerized systems. The aim of our project was to provide the SNCF (today for delegated infrastructure manager, and tomorrow for rolling stock departments of railway subcontractor) with an operating method for the formal validation of critical computerized systems, especially for the Interlocking and ETCS/ERTMS systems. A formal proof method by assertion is presented in this paper; it covers the specification and its software implementation. With the proposed method and its associated tools we verified that the system follows all safety properties all time and does not show superfluous conditions: it replaces the platform checks and is in accordance with the existing SNCF testing procedures. The advantages are a significant reduction of testing time and of the related costs and an increase in the test’s cover rate (deterministic safety vs. probabilistic safety). The paper assumes that the formal methods mastery by infrastructure engineers is a main key to prove that, during the life of the system, an increase in safety is not more expensive.

Keywords: interlocking, ERTMS/ETCS, formal proof, formal validation, safety analysis

Introduction

Transport history is full of exemplary accident cases and disasters of any type. For this reason principles and design procedures progressively appeared, thus creating, formal validation methods [1] and exploitation rules in order to reduce and to prevent the risk of an accident. These enabled spectacular progress. The development of computer science and industrial automatism was, and still is, a source of increasingly efficient new solutions. But they are also a source of new complexity that increases the difficulty of a durable and economic design system, as well as the evaluation of transport system safety. The complexity of the system and the size of software have increased for different reasons, mainly the rapidly increasing power and memory capacity of computers and the demand by suppliers and customers for more and more functionalities. Software is immaterial and as such does not seem to have limit. These days, the majority of sensors and interface devices contain software. Software has become critical because it is used today to support safety functions. These functions can, in case of fault (error), lead to fatalities, financial losses and property or environment damage.

Today specification requirement modes are treated in the European standards but they are used by the great majority of suppliers as a list of requirements outside the context of use and without defined limits of the system. This leads to respecting a “means of obligation” (non deterministic safety approach reposing principally on the respect of procedures) and to leave to one side the “safety results obligation” (deterministic approach of safety, a minima in the definition of the system functionalities). It is a question of reducing the prohibitive expenses of the manual activities of critical computerized systems for verification and validation checks in the particular steps of adjustment (correction of anomalies) or migration.
Challenge H: For an even safer and more secure railway

To keep the French signalling systems at its high level, SNCF Infra has developed a “generic fail safe automatism” concept. The functionalities of the system (described in an “Open Model” for example) are in their context of use formally provable. This project led to the development of specific design constraints and the deployment of computerized interlocking systems such as PIPC in order to make formal proof possible [3]. In this paper, a special focus is brought on the method developed and used by SNCF for reducing delays and costs with, at the very least, an increase of safety level in the validation of interlocking systems. Our ultimate aim is to define a generic framework supporting a formal proof of SIL4 functional software. In the scope of this paper, we present the French interlocking system using functional Petri nets specification. The formal validation method is based on these Petri nets.

The Problem

Experience shows that as computerized systems become more complex; cost pressures and delays grow. For example tasks corresponding to the integration tests and validation are more and more expensive. The technical railway characteristics of computerised systems enable the identification and anticipation of potential conflicts and impose putting safety in the centre of all actors’ actions. Their synthesis enables the construction of a very safe system.

- Any modification of a component of the system can have an impact on another component and compromise the safety of the whole system. For example: The human being limits are attained and exceeded concerning technological complexity and design, without taken into account testability constraints. Software anomalies are today considered to be "normal".
- Standards and actual practices, in particular for critical computerized systems, define means to be respected and don’t guarantee that the safety result will be reached,
- The industrial world tends to differentiate no more expected functions and means to accomplish them, the all is inserted into the “software”,
- The software anomalies correction does not drive towards stability but towards new anomalies some of them critical…

Therefore it is necessary to look after the lifecycle at all system stages. The design path necessarily drives to a probabilistic safety approach because software can no longer be tested in an exhaustive way. It is necessary to be aware of the consequences of a statistical approach when it comes to validation of the functionalities of computerized critical systems. Experience shows us [7] [8] [9] [10] [17] that conventional methods cannot be applied to new and complex computerized signalling systems.

When it comes to the legal framework, the infrastructure manager still remains the guarantor of the safety of trains and customers.

Safety on French railways is historically based on a deterministic approach. Any effect has a reason. If an effect is undesirable, abolishing its causes must enable one to avoid it. It is a question of making sure that such an event never happens. This principle must also apply to computerized systems when an unsafe fault is intolerable. This is possible only if measures are taken at the design level. Superfluous conditions limiting the operational activities and computerized systems safety cannot be estimated by probabilistic approaches. Experience shows that the main part of the critical systems risk comes now either from stochastic faults, or from systematic faults (Figure 1). These faults affect the whole system (for example systematic software) because faults occur due to design errors, faults linked by dominoes effects, etc… This aspect reposes today, with the known limits, only on «insurance quality» procedures imposed by the standards.

Figure 1: Faults having led to an accident are principally linked to specifications and to maintenance (Source: Gartner Consulting 2009 [21] [22])
If there is a specification error (a wrong interpretation of specifications by the software designer or a case not treated by functional specifications for example) whenever the non envisaged inputs sequence occur, the system leads, in a deterministic way, to an unsafe state of outputs. This is independent of the hardware architecture and SIL system level. It is a systematic defect which will produce a dangerous situation in a deterministic way.

It is therefore necessary to differentiate the probability of obtaining the not envisaged inputs sequence and the determinism of the treatment accomplished by system functionalities. These errors of comprehension can lead to a common failure, where all the calculators make the same error at the same time. Contrary to faults due to hardware defects which appear in an unpredictable manner, the consequences of functional design errors cannot be prevented by redundancy. An unsafe specification error "awakened" by a particular input combination then irremediably affects outputs whatever the number of calculators in redundancy. In certain cases, one feared combination occurs once or its unsafe effect can be memorized without alarm. Tests generators give no guarantee as to the exhaustiveness of the performed tests. Their cover rate is restricted due to:

- The temporal impossibility of accomplishing all envisaged situations,
- The difficulty in building tests to prove the functionalities respect in “not envisaged input sequences” for specific contexts of use if these were not considered during earlier specification studies (i.e. “Do we think to test what we forgot to specify?”),
- The possible “not determinism design” of any system...

A safe system is designed and built with simple principles in order to be exhaustively validable. It should also create the conditions of a formal validation of the functional software functionalities.

The option chosen by the SNCF for PIPC design is to clearly differentiate, on the one hand functional software and on the other hand hardware and basic software. This allows the industrial application of a formal validation method concerning the functional software. Let us note that all these points are true for complex architectures of the modern railway systems such as ERTMS or CBTC.

System and over-system: The functional validation of a computerized critical system must take into account the over-system in which it is going to operate and to interact. The system has to be defined in terms of procedures, actors and interfaces, on the infrastructure or onboard side. The over-system defines the framework in which the system under definition will have to be proved (physical, temporal and regulation framework). It is, for example, the case of an interlocking system to define:

- What is the type of the blocking system? Permissive or not permissive, if so with what are the permissiveness rules?
- What procedure has to be applied in the case of the absence of control of an unchecked switch? Can or cannot the driver exit his locomotive to check the position of the switch and if needed move it locally?
- What is the route automatic destruction mode? Destruction takes place in “rigid transit” (all switches are free at the same time) or in flexible transit (switch by switch are free according to their liberation), if so with what limitations?
- What is the route manual destruction procedure? Is manual destruction allowed, if so it requests a specific delay and an operator double order?
- What is the transit interlocking release procedure? Is the unconditional release order accomplished on the field, with the approval of the control centre, by the driver or directly by the control centre?
- Is there a sufficient sliding distance between the protection signal and the switches to be protected in order not to request an “Overlap”? Etc…

Obviously, answers to these questions differ from one country to the other (they will differ in spite of the advent of the new moving block systems) and translate the fact that safety principles, exploitation rules, as well as the environment and context of use differ according to countries. The same interlocking system can therefore be considered as "safe" in one national over-system and as “dangerous” in another one. This shows the importance of the “not ambiguous interfaces definition” with the over-system (Figure 2) for the system to be validated.

1 French mechanical interlocking systems enable the realisation of exhaustive tests, notably due to the fact that if two levers are in correlation, it is only possible to move one at the same time.
Challenge H: For an even safer and more secure railway

Define the safety properties of the signalling system

Figure 2: Relation between System –over-system and formal validation possibility

The safety properties of the system and the environment functioning postulates of the over-system define the validity domain of the exploitation rules of the system. They depend on the over-system and its system interfaces. Let us note that for a SIL4 level it is necessary to exhaustively define the domain allowed by every interface with the over-system to be able to hope to accomplish an exhaustive system validation. Two cases are to be considered:

- The development of a critical computerized system in the context of one given over-system;
- The introduction in a given over-system of a system defined partially without any given over-system.

In the first case, the system functioning principles are inherited from the over-system. The system is designed in concert with the regulation over-system principles: the validation of its functionalities is possible. This is the case of the computer interlocking PIPC which we are going to introduce later.

In the second case, the functioning principles are completely or partly different from the over-system. The over-system must then be adapted to guarantee a safe functioning of the system and of the entire over-system. Experience shows that in this case the validation of the system functionalities is more difficult or in general not possible. The use of formal methods for functional validation and specification is today necessary for the definition of every new computerized signalling system. Otherwise, test ratios cannot be absolute leading to assume putting into service a signalling system knowing it has systematic and certain unsafe defects.

Interlocking system: An interlocking system enables the control of switches and signals: its mission is to manage the rail traffic inside a geographically delimited area [4]. Train management consists of forming the train routes and providing every train the correct movement instructions by taking into account the state of other routes and the positions of other trains. These actions are decomposed into basic operations which must be performed safely to avoid collisions and derailments. To anticipate the constituent and/or computer unit renewals it is necessary to clearly define technical and functional interfaces (Figure 3) to allow an easy constituents replacement [3] [4]:

- Interface 1: between the signalling installations in computerized technology and those in the field allows regenerations of the one or another one independently;
- Interface 2: between the physical support (computer architectures: hardware and basic software) and functionalities requested by the track plan and exploitation requirements.
In order to reinforce these interfaces two cases are considered. Both are treated separately in the current French interlocking system:

1. The functionalities that have to be fulfilled by the system and that have to be translated into the programming language of the computer system. In this case, one must question if the specifications and their transformations into the acquired final code are 100% correct? It is clear that putting into service an installation with a high safety level (SIL4 for instance) involving “a” functional incorrectness that persists in the final code, will necessarily lead to an accident after a certain time in a deterministic way. Experience shows us however that this scenario exists [1];

2. The architecture support (hardware and basic software) that allows the realization of the system functionalities with a safety level (tolerable hazard risk fixed by safety standards). In this case one must question the level of maximum residual error that can be guaranteed by the architecture under real usage conditions? This concerns an unsafe failure rate per hour involving the possibility of an execution error of the system by including active modules and their real-time constraints, the data and control flow, synchronizations, connection with the material...

The number of functionalities of every interlocking system is different. Each can be formally proven before being put into service while executing on the same target machine. To prove the functional software is equivalent to accomplishing the proof of the software executed by the target machine. Therefore there is no difference between the high level model and executed out code. The formal method chosen can be inserted in the actual studies/tests process. The proof brings insurance that the system specifications in its environment (over-system and interfaces) do not contribute in a deterministic way to the elaboration of a system identified risk.

For the reachable system state and formal proof aspect, the following property is ever respected:

\[ \text{VE}[i^{\text{th}} \text{ injection}] \leftarrow \text{VE}[(i-1)^{\text{th}} \text{ injection}] \rightarrow T(\text{event} | \text{condition}) \]

That mean that the system state vector after the \( i^{\text{th}} \) event injection is only defined by the previous system state and the transition \( T \).

For the maintenance engineering, proof also aims at:

- reducing expenses on the lifecycle of interlocking systems (development, functional evolutions, material obsolescence);
- supporting the safety level of interlocking systems in spite of the functional complexity increase and of the actual qualification processes limits (safety based on the development quality, on tests scripts with restricted cover rate);
- defining a unified formal validation method of the functional expressions. This method has to be used in practice by signalling engineers without highly educated employees in mathematics (can be also used in the framework of a offer to tender);
- supporting the competence level of the signalling people (signalling studies, tests and maintenance).

It is not a question of formal software design, but of formal specification and validation of functional software designed by signalling experts: it is a validation of the safety properties in the environment framework, while using postulates (procedures, operator’s action…).

**Formal validation of the interlocking system functional software**

**AEFD Petri Nets Language**

The AEFD language (Deterministic Automat with Finite number of States) [5] is a description language of the system functionalities in form of a finished states graph network\(^2\). The graphs represent the signalling functionalities of the interlocking. Every graph can be only in a finished number of states and can be only in a single state at a given moment. Every graph state is represented by one and only one place. The graph states are linked up by transitions. A transition is composed of an events equation, of a conditions equation and of an actions list. Elements defined in transitions are called functional entities. They are used for the identification of the external physical elements and of system internal variables. Every functional entity is a Boolean variable of which the type is determined by an identification prefix:

\(^2\) Automate Concurrentiel à Contraintes (ACC)
Challenge H: For an even safer and more secure railway

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Functional entity type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTL</td>
<td>Control of system input (reading the state of a safety relay…)</td>
</tr>
<tr>
<td>CMD</td>
<td>Order of system output (order of an external safety relay)</td>
</tr>
<tr>
<td>ACT</td>
<td>Internal graph activating (use for synchronization between graphs)</td>
</tr>
<tr>
<td>IND</td>
<td>Indicators (internal variable) or result of numerical calculation</td>
</tr>
<tr>
<td>MSG</td>
<td>Message (order sent by an external system)</td>
</tr>
<tr>
<td>DTP</td>
<td>Starting of time delay (order)</td>
</tr>
<tr>
<td>FTP</td>
<td>End of time delay (event)</td>
</tr>
</tbody>
</table>

On a given place of a graph an indicator always takes the same value. The indicator Boolean value can only be changed by the transition of one graph. All the graphs can achieve the value of any indicator to use them in the event or condition equations of its transitions. Possibilities of use:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Event</th>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTL</td>
<td>X (if logical state change)</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CMD</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ACT</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>IND</td>
<td>X (if logical state change)</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MSG</td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

The functionalities are realized by communicating graphs. It is precisely of:

- states machines: Petri networks such as every transition is linked up with exactly a place in entrance and a place in exit by arches of value 1,
- not autonomous networks: it is a class of Petri networks on which evolution does not depend only of the network state but also the linked environment states,
- Petri networks having a binary marking.

The state change of a graph is always linked to an external change. Particular attention must be given on limitations of the existing interpreters tools of Petri networks. The classical interpretation rules quickly drive to a «combinatory explosion» and/or to a «non determinism execution» if they do not watch out for it. The choice of Petri networks is not to call into question. It is necessary to identify weaknesses of these interpreters:

- The non differentiation of external and internal events (an event activates the graph A which activates the graph B and so on). The exploration puts then on an equal level a new external event and an internal event between graphs A and B, creating conflicting systems states;
- The dependency with the graphs writing order. The transition crossing leads to the marking updating. This updating can then change the crossing conditions of other transitions which will be assessed later (higher rank graphs). The modification in the order of graphs interpretation, the modification of the valuation priority or the insertion of a new graph can change the interpretation of all graphs. It is necessary to differentiate for a deterministic graphs valuation the future and the common places marking. It is the reason of the "indicators" creation;

It is for these reasons that we defined the AEFD language: it uses the formalism described in the previous paragraph which differs in comparison with classical Petri nets by:

- Graphs writing in form of a 6 lines text file (graph name, start place, goal place, events equation, conditions equation, actions list);
Challenge H: For an even safer and more secure railway

- Introduction of the indicator notion valuated after the execution of the marking changes;
- No intervention of the time does in the graphs;
- Introduction of the notions of external and internal events.

Any design rules have to be respected during the formalization of the functional graphs (software). They guarantee a correct graphs interpretation, reduce the number of system states (without artificially system states) and reject any indecision possibility in the graphs interpretation. Let us specify the characteristics of this AEFD language. A graph is constituted by places and transitions linking the places, conditions of transition crossing, actions linked to these transitions and place marking. A "token" occupies the place in which is the graph.

**Computerized Interlocking Module (MEI)**

The design choices of the interlocking module have to guarantee a deterministic interpretation and a not ambiguous understanding of defined AEFD graphs. The interlocking module of the PIPC interlocking system was deliberately designed to insert necessary properties for the subsequent realization of a formal proof:

- The functional specifications (asynchronous Petri nets in AEFD language) are directly interpreted, without being rewritten;
- A unique external event is treated at the same time; the chronology of external events is so kept in any circumstances;
- The time delays management rules are defined out of the graphs.

The resolution engine assures the dynamic execution of functional graphs. It consists of:

- The external events management (CTL state change of system input, MSG reception of an external message, FTP expiry date of a time delay, ACT activation of a graph by another one);
- The internal events management resulting from the transitions crossing (IND positioning of an internal variable, ACT activation of a graph);
- The event treatment (internal or external);
- The management of the internal variables manipulated in graphs:
  - the reading of the internal variable common state;
  - the positioning of an internal variable in a desired state;
  - the detection of the state changes of internal variables.
- The activation of processes needed by the actions treatment:
  - the activation of an internal variable (IND: injection of an internal event),
  - the activation of a graph (ACT: injection of an internal event),
  - the activation of an output (CMD),
  - the starting or stopping of a time delay (DTP or ATP).

The activation of an internal variable is postponed until the end of the current event treatment. The action is memorized in a FIFO of actions postponed before being accomplished. The detection of a state change leads to the injection of corresponding internal event. The resolution engine is designed for events; a transition can only be crossed on detection of an external event. For every injected external event, the graphs activation for the event treatment is done. The order of events valuation is always the following one: treatment of an external event (first external event memorized); treatment of all internal events resulting from the treatment of the current external event; treatment of internal events generated during this treatment; treatment of following external event, etc. During the event treatment, the performed treatments are:

- Research among all the graphs of transitions potentially concerned by this event (all transitions for which this event is originally);
- The check on their marking on the origin place;
- Research among these transitions the crossable transitions;
- The valuation of the crossing conditions for each of these crossable transitions (or if there is no condition):
  - Activation of applicative software linked to the transition crossing (actions undertaken since graphs);
  - Actualization of the marking due to the crossed transition.

The valuation of transition conditions corresponds in a Boolean True if the crossing conditions are verified. It provides the status of crossable transition to the potential transition in the course of valuation. Graph conditions are exclusively expressed with “products of sums” or “sums of products” of Boolean variables. In
that way, the interpreter allows to get a deterministic functioning of the system. The interlocking module includes fundamental properties to apply our formal method:

- It keeps the chronology of external events, whatever the time slice between these events. The events are taken into account in order;
- It treats one external event at the same time. This one is "spread" in the graphs which constitute the functional software. Once ended internal spread, the following external event is then taken into account;
- It interprets graphs without any rewriting or transformation;
- It includes fixed interpretation rules; the transitions equations are read in a defined order;
- It acts as an abstracted machine with no commutation time (all transitions are instantaneous).

**Principle of the Formal validation method**

Used within the strict framework of the operating rules of the MEI, our formal validation method of the safety properties [2] [5] [6] covers the whole development cycle from the specification to the execution, and rests on a mathematical proof. The method is a kind of “Model Checking” which would not apply to a “model” but to an interpreted target code. We don’t carry out scripts of tests but we check exhaustively that the system respects all the safety properties and the awaited functional properties, and does not present superfluous conditions. With this intention, we use the fact that the interlocking module is strictly deterministic and that the functional one is interpreted in real-time the number of states of the system is finished. It is not a question “to model” the functionalities awaited (with an unknown distance between the model and the real execution, between the fact of modelling the coded execution of the application and of carrying out in real-time) but to work directly on the specifications carried out in the target machine. The writing of the properties of safety is also direct in language AEFD. It can be realized by the railway engineers. The fundamental idea is: “if a property is true in an occupied state and if the conservation of this property during the transition which follows this state is guaranteed, then the property will be true in the new occupied state. The demonstration can be continued as a long time as the property is preserved”. This principle is used for the properties of safety.

![Figure 4 – Formal validation principle](image)

To make a formal proof of these automats amounts to check for each state reached a list of logical properties and to answer the question: Does the design respect the specifications? The proof rests on the detection of safe states, the use of mutual exclusions with unsafe states. The proof rests on analysis of the reachable states from the system and can be summarized by:

\[
\text{Post}(\text{VE}[0]) \cap \{\text{VE}[i] \text{ unsafe}\} = \emptyset
\]

The expression shows the fact that the intersection between the whole reachable states from the system and the whole unsafe states is an empty set. Then the proof that the system cannot reach one of the states unsafe is assured. But if the opposite occurs the proof is rejected. The definition of the unsafe states is carried out by particular graphs, carried out in parallel of functional reality, named “proof automats” (PA).
Challenge H: For an even safer and more secure railway

- When a property is valid during the current state of the automat (system state), it can be the subject of a proof automat in two places. When the feared situation has suddenly occurred the automat must take a particular place and the US indicator (Unsafe Situation) is positioned in True;

- When a property should be valid only when the state running of the automat belongs to a part of the states accessible from the automat, it can be the subject of a PA to more than two places. When the state running of the system belongs to the target part, the PA takes a particular place activating the transition representing the property to be checked. At this time, when the dreaded situation occurs the proof automat takes a particular place and the US indicator is positioned in true.

The a priori knowledge of the interlocking functions of the station, for each operational phase, makes it possible to define proof automats formalizing the safety properties to be respected for each one of it. Taking into account the station characteristics it is possible to retain the valid PA with the values corresponding to the route to prove.

The writing of the proof obligations (safety properties, superfluous conditions) is done by the signalling experts. Experience shows, that it can not reasonably be produced by RAMS engineers.

By the analysis of the topology of the track plan and the technology retained for the shared resources (track circuit, trackside equipment...), the PA make it possible to check additional properties: properties of non superfluous conditions and those of non respect of the operation postulates.

Application of method in the case of an interlocking system

This method was applied to several real interlocking system of type PIPC, by means of tools automating its execution. The functional graphs were proven formally. For example, the station of Nurieux is a station of 16 routes put into service at the end of 2009. Figure 5 presents one part of the station... A double click on each graphical object reveals a parameter setting windows.

![Figure 5: Track plan of the Nurieux station (abstract)](image)

Figure 5: Track plan of the Nurieux station (abstract)

The safety requirements relative to each route are identified by the agents in charge of the validation of the station. These parameters are:

- do the functions of interlocking attached to the route according to the awaited functional program of the station exist.

- is the parameter setting of these functions according to topology, of the real distances from establishment of the resources (signal, switches, sensors...).

This work is independent of the signalling study process and in particular without influence of the choices operated in this process. The elements are gathered, visualized and validated by these agents on the basis of information turned over by the tools. This information automatically allows of instanced the generic proof automats required by this route, the useful postulates for the explorer and the prover (Figure 6). The treatment of a route will be used to us as concrete illustration of the procedure, the treatments and the interfaces of the users.
Challenge H: For an even safer and more secure railway

The implementation tools of the method were used to treat real interlocking system like shown in the Figure.

Figure 6: Generic proof graph for a route

Figure 7: Formal validation tooled process at SNCF Infra
In Nurieux station, the application of a route (3425-3431) will serve as a concrete illustration. The number of explored external entries changes is about 250,000, of which: 25,000 of them were followed by valid transitions (functional condition and safety property: truth), 117,200 were rejected by the functional (condition false), 7,000 distinct functional transitions were discovered by the two exploration methods (with the same result), and finally 920 state vectors system were provisionally memorized before new injections. The control screen of our tool allows a signalling engineer to follow step by step, if needed, the chronology of internal and external events and states changes of graphs. The analysis allows the engineers to discern the origin of a possible gap between observed functioning and waited functioning.

**General implementation of the method**

The method introduced is used by SNCF Infra to specify the functionalities of a new system to be developed and to obtain necessary elements to assure the future system integration. According to the supplier’s position, the following two functions are in general followed. In both cases responsibilities and industrial risks of the infrastructure manager and the supplier are clearly identified. The signalling functions are:

- **Encoded** in the target development language. Development takes place in the usual conditions, with inherent difficulties in validating this type of application, and

- **Interpreted by the target machine.** The supplier understands the “target structure” (with hardware and basic software) and is able to interpret with the requested safety level the functional specifications given by the infrastructure manager. The target machine can also be reused without additional development for other critical applications. The infrastructure manager possesses the ability to modify functions of the critical product without resumption of all safety demonstration (hardware obsolescence or supplier failure) and so to assure the system integration of the product in its over-system increasing the life time of the system.

This second approach (target machine) is the most interesting over the long term for both parts. The product can be re-designed, without modification of its basic functionalities. Signalling functionalities can be changed, without the need for the product and the target structure being modified. The simplicity of the description language, its independence from the computerized target structure, and its comprehension by the signalling engineers allow it to:

- Guarantee the quality of the primary work of good design, in particular the increasingly complex systems, including, those based on computer technology and programmed with software,

- Improve the management of the interfaces with operations and human factors,

- Reduce the amount spent on the safety assurance of signalling systems,

- Reduce the lack of common understanding between signalling engineers and software supplier’s team. This is currently leading to confusion and duplication of effort,

- Reduce the misinterpretation of the requirements of certain standard and legislation, and inappropriate use of some guidance regarding the requirements for the independence of assurance arrangements,

- Give the infrastructure manager the possibility to specify exhaustively, considering the over-system (operations, environment, human factors, national principles…), the signalling functionalities of a new system: it can then be considered as a “functional white box”. This point is particularly important for the infrastructure maintainer and responsible for the maintenance engineering,

- Reduce the safety assurance costs after each functional evolution in the life of the system. The functional description has only to be formal proved and will be interpreted by a validated target structure.
The first approach leads to:

Figure 8a: First use of our way of specification and validation of the functionalities of a critical module

The second approach leads to:

Figure 8b: Second use of our way of specification and validation of the functionalities of a critical module
The systems obtained at the end of the design are different. In the second case the functional software could be considered as an “Open Model” given to all the potential signalling suppliers in competition for a new given system. This method could be applied on ERTMS/ETCS applications (RBC or EVC) with beneficial contributions.

The process aims at separating clearly «how to make» (hardware and basic software / supplier) and «why and what to make» (instanced functionalities for a track plan and an operation program / infrastructure manager) and at accomplishing a formal validation of the signalling functionalities (by taking into account the over-system).

Conclusion

The work carried out gives an applicable original industrial method for the validation of the achievable functional specifications of an interlocking system. With this applicable original industrial method for the validation of the functional specifications of an interlocking system, it's possible to prove automatically that the specifications carry out in a safety way the entire awaited requirements and that the product is correctly designed. Formal AEFD language can be used to formalize the specifications of new comparable product or interlocking system. The method is applicable while raising usual criticisms to the formal methods. The users can not be skilled any of the mathematical techniques. The formal approach makes it possible to reduce the design and maintenance costs of the critical software. This in turn allows Industrial tools to be developed and put on the market next time.

Our work builds a bridge between the university and industrial worlds, in the mutual interest of both communities. The university indeed has many theories, and methods relative to the “automats with finished number of states”. The industrial world must design, maintain and make evolving many critical safety systems, computerized systems where all the functionalities can be specified by automats with finished number of states. Our approach shows that it is possible to realize physically, within the constraints inherent in the system SIL4 development, a target machine behaving exactly like an automat with finished number of states. The functional “model” passes from the statute of “abstraction of reality” and “specification to be realized” to that of “real description” or “specification carried out” in real-time.

It was a question of proposing a formal method which allows, in addition to the awaited economic and safety gains, of ensuring the success of the application of formal methods in the railway field and the skills conservation of the railway agents. The infrastructure managers can thus specify the computerized systems in AEFD language and the two technical interfaces I1 and I2. Language AEFD makes it possible to obtain the insurance that the specifications are correct in the context of use, that the postulates and the safety properties, unknown factors in detail by the suppliers, are well filled and that the infrastructure manager has the means of controlling the product throughout its lifespan. This goes in the direction of a global reduction of the maintenance and development costs, without reduction of the safety level. Thus, in addition to the initial contribution of the formal methods in term of safety, the formal methods contribute largely to the economic assessment of computerized critical systems.
The method and the associated tools developed by the SNCF were the subject of a patent. The described steps are used from 2011 to validate formally the new interlocking systems having to be put in service on the national network or being the subject of modification.

The ERTMS defines the standard for interoperability between the onboard train protection systems (ETCS) and the railway infrastructure. This standard is decomposed into several documents (named Subsets) focusing on specific parts of the system. The AEFD language could be used to model the EU and the national requirements expressed for the RBC. One of the results of the method application is to replace the tests specified in Subset 76 (not exhaustive test cases). This modelling effort is aimed at generating an AEFD interpretable model according to the customer’s target’s language and providing the required artefacts to prove the match between the generated “Interpretable Model” and the ERA and the national requirements. The customer can also use the model without code generation. In that case, the AEFD Model can be used to verify the match between the model and the ERA requirements completed by the national requirement.

With the actual software design without formal method, the application of the CENELEC standards as EN 5012x (x = 6, 8, 9) for new safety applications produces a “probability” that certain (unsafe) failure rates will be achieved. There are many reasons to apply formal methods and they present advantages at several level of the project lifecycle: at the time of writing an invitation to tender (responsibility of the owner: which system in coherence with the over-system?), at the time of transferring the expression of functionality at a supplier (what to do by the system?), at the time of designing the equipment by the software designer (how to do it?), at the time of checking before putting safety signalling facilities into service (responsibility of the system integrator).

In the context of greater economic constraints and increasing complexity of computerized tools, formal method can really and in concrete terms facilitate the system integration in its over-system in order to master costs, delays and RAMS performances for new and complex systems.

References

Challenge H: For an even safer and more secure railway


