Formal Methods in Industry: The State of Practice of Formal Methods in South America and Far East

Aryldo G Russo Jr.

AeS Group & Research Institute of State of São Paulo (IPT),
agrj@aes.com.br

Abstract. The use of formal methods has constantly increased, although with basically two constraints: their use has been concentrated mostly in Europe, their Mother land and they have been used only by big companies which are in charge to develop some kind of safety critical applications, what, in a first look seems correct. The aim of this paper is to present the usage stage of formal methods in other parts of the world, mainly South America, and Far East. A personal comparison of some formal method tools, namely: Atelier B[1], RODIN[2], and SCADE[3]is also presented. The comparison methodology is based on three different points of view: capability, I mean, how these tools can satisfy project constraints, usability, basically, what's the difficulty the user faces when trying to use the tool, and adequacy to the current development process.This work describes also real applications in industry, sometimes not the formal method usage itself, but how the formal method culture can drasticaly helps on the development process. Finally, some of the gaps in industry wishes that could be fulfilled by some applications are sorted.

1 Introduction

The primary objective of this paper is to present the curent State of Practice of Formal Methods in coutries outside Europe, namely, Brazil and Korea. In this sense, I would like to present it as the utilization of formal methods in general, and, moreover, not only the application of one method or another, but how the principles that guide the formal methods usage can help in the software development process.

But, before talking directly about the subject of this paper, I'd like to give some background information about the reason I started to work with Formal Methods, and my involvement in academia. Then, I will present a general scenario of how these methods are being used nowadays in the places I meintioned before. In the remaining of this paper, I will present some industrial areas where we can find already some use of formal methods.

Finally, I will present a comparison of three tools, namely, AtelierB[1], RODIN[2] and SCADE[3]. This comparison is based on three aspects, tool capability, usability and adaptation to the current development process. I will show also, some real application of these tools, and the work that was performed to change the way that industry was used to think about software development, even in safety critical areas.

At the end, I will present some gaps that, from my personal point of view, can be fulfilled with some new or in development phase, plugins and language extensions.

1.1 The AeS Group

The AeS Group has developed railway sub-systems since 1998. Among the systems developed by the group, the door system became one of the most important in the railway market due mainly to the architecture used (modular, and with distributed processing) and, since this kind of system deals with human lives, the strong concern of the group with reliability and safety.

During the development process, four versions of the main controller (called CGP) were created and, at each iteration, additional safety features were incorporated, using different techniques, such as hardware redundancy where different sources are employed to activate an output (for example, some safety outputs have to be activated both from a software command and from an external solicitation). Safety and reliability studies were performed, and all identified potential weak points that these studies revealed were mitigated to prevent, or at least minimize, the hazard effects. In the current version of the equipment, even applying all these hardware techniques, safety issues remained to be fulfilled, as well as the software(firmware) correctness, robustness and failure avoidance.

From that time, and after facing several pitfalls, AeS Group is becoming more and more known as a company that has the needed know-how to develop safety critical application, and, nowadays, it's been in charge of several training sections around the world teaching software development process for safety critical application based on a formal method mind.

Due to the advances in technology, many safety functions that were handled by hardware are now responsibility of the embedded software. This fact triggered motivation to use formal methods in standards relevant to software safety [4]. Some standards can be followed to increase the safety level of an equipment. One of them is the IEC 61508 [5]. This standard presents four levels of safety, the so called Safety Integrity Level - SIL, and above level 2, a formal specification is required or suggested to achieve a certain level of completeness, robustness, and safety, that grows as the level grows. The goal of using formal methods is to produce an unambiguous and consistent specification which is as complete, error-free and without contradictions as possible, however simple to verify.

To address the group concern with safety, the AeS group decided to identify a formal method that would best fit the current CGP SIL 3-level requirements and railway industry standard practices and standards (as is the case of CENELEC EN 50128[6]).

With this in mind, and taking into account restrictions such as, mainly, the size of the company (only 15 employees) and the lack of deep knowledge of the method itself, the AeS group decided, first, to study and use the B method[7] and, second, to look for assistance from academia, which was obtained from two Brazilian Universities (Universidade de São Paulo and Universidade do Rio Grande do Norte).

Nowadays, AeS Group has also the support of DEPLOY project and some universities like University of Southampton, and University of York. Of course, AeS Group is also supported by companies like ClearSy and Esterel.

1.2 Research Institute of State of São Paulo (IPT)

With the base of previously performed studies, but with some reluctance, I decided to finally initiate a "formal" dedication in the Formal Methods field, and have chosen the Technological Research Institute of State of São Paulo (IPT) as my starting point. During the last years, some articles were developed at IPT but the relationship with other research and academic centers was the main incentive to study the application of these methods in real world systems.

Meanwhile, I joined the Software Requirements Specification Laboratory (SoftREL). The main goal of SoftREL is to create, deploy and disseminate a research environment for post-graduate IPT students and other researchers, helping them to develop academic research and artifacts related to software requirements engineering. The laboratory intends to develop academic and industrial partnerships aiming at the development of engineering techniques and tools required to deliver more reliable computer systems.

1.3 General scenario

In order to picture out the differences in formal method application outside Europe, I will give you some information about the current software engineering process that is being applied at this moment in the process of safety-related application development.

Basicaly, the software development process presented in the IEC 61508[5] is well known in South American companies, but as the time to market is, normaly, extremaly short, those recomendations are put aside, and the craft process is followed. This process is basically the reception of the primary specification, the coding phase is made relying on the personal expertise, and the tests are performed as few as possible. It's already a good scenario to use formal methods and try to better the process without changing the manual tasks.

Talking about Far East, those process are barely known. As presented in [8], the adoption of the recomendations referenced in the software development process are in it's infancy phase, meaning that even the standard understanding are not clear enough.

Taking the "V model" reference presented in figure 1, it's possible to point out:[1]

– Companies in South America

[1] those points are based on author's feelings

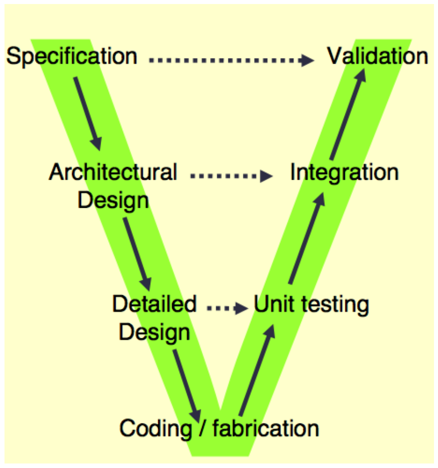


Fig. 1. V Model - Software Development Model

  - They are aware of the whole process
  - They usually rely on tests to guarantee the expected behavior
  - The transition takes place directly from NL specification to the code phases, some times, through an intermediate phase, based usually in UML specifications
  - there is no apparent traceability methodology
- Far East - South Korea
  - The coding phase starts right after receiving the specification
  - Quality is the main concern, but no defined process is used to ensure that
  - They rely on experienced professionals to reach the desired quality level
  - Clearly, there are only three phases: specification, coding and integration tests

Based on this view, it's crystal clear that is not possible to go directly to the pure application of formal methods. First, it's necessary to create a better culture of software development process.

But, what happens if, while doing so (creating a culture) we could, in small doses, integrate the formal thinking, and show the advantages in using those methods to, in the far end, speed up the development process, and decrease the costs of software development (or, if you think it's easy to understand, to minimize the so called ”non quality costs”).

This is the current scenario.

2 Where formal methods (could be) are used

Many different industrial areas, where safety and reliability issues are highly important characteristics, have been using, or at least have tried formal methods in order to increase their confidence that those requirements are met. Those industries are, mainly, Nuclear[9], Medical devices[10], Avionics, Aerospacial and transportation [11]. Some examples are the emergency contention measures in nuclear power plants, health support devices in medical applications, automatic pilot on avionics, positioning systems in aerospacial and signaling systems in tranportation just to cite a few.

This means that there is plenty of space for the adoption of supporting tools that could help either the development process (either system or software) in the sense of automatizing some parts of it, and also, in some cases, for speeding up those development tasks difficult to perform, while the developer uses his efforts in other more conceptual phases.

Unfortunately, even if the referenced industrial areas are something that exists all around the world, the application of formal methods is not the true reality in American and Far East companies that work on those fields. It's something not easy to explain, as, in theory, the standards that should be followed by all those companies are the same, as it's the case of IEC 61508[5](related to general functional safety, it means, not field specific), DO-178b[12] for avionics and EN50128[6] for railways. All of these standards highly recommend the use of formal methods either in the specification phase or in the design phase in order to achieve high levels of the so called Safety Integrity Levels[13].

In order to change this scenario, the distance between mathematical notation and the normal procedures used so far has to be shortened, and for that some highly desired characteristics should be included in the current tools in order to reflect the activities that are normally performed in those industries.

Fortunately, it might not be so difficult as, at least, the development model that has been adopted in those industries (V model1) is not different from the model used in a formal model development.

The focus of this paper will be in railway field, as it is my area of application. Taking this in consideration, as can be seen in several different works, like [11] and [14], the B formal method is the most used one in this field. Recently, the (fromely lustre) Esterel[15] formal method began to be used as well, and the support tool for this method, SCADE[3] was certified as capable to produce safety code up to SIL4.

In spite of the field of application, formal methods, and it's related tools, can help in the development process replacing the human interaction of the phases (see figure 1): Detailed design, coding and unit testing, by an automated process, and by doing that, can help to speed up the development process and better the "quality" of the final product.

2.1 Formal methods and tools

for the sake of good understanding I present this brief sections about the formal methods meintioned before.

B method The B method for software development [7] is based on the B *Abstract Machine Notation* (AMN) and the use of formally proved refinements up to a specification sufficiently concrete that programming code can automatically be generated from it. Its mathematical basis consists of first-order logic, integer arithmetic and set theory. Industrial tools for the development of B based projects have been available for a while now.

A B specification is structured in modules which are labeled according to their abstraction level: MACHINE, REFINEMENT or IMPLEMENTATION, from the most abstract to the most concrete. The model of the CGP is at the machine level. In the B method, machines are to be proved consistent with respect to some specified properties (particularly, the invariant of each machine).

One of the main parts of a B module is the state space definition, which appears in the VARIABLES and INVARIANT clauses. The former enumerates the state components, and the latter defines restrictions on the possible values they can take. Essentially, if V denotes the state variables of a machine, the invariant is a predicate on V. Let us denote INV such invariant predicate. All verifications carried out throughout the development process have the intention of checking that no invalid state will ever be reached as long as the operations of the machine are used as specified.

For the specification of the initialisation as well as the operations, B offers a set of so-called *substitutions*. The semantics of the substitutions is defined by the *substitution calculus*, a set of rules stating how the different substitution forms rewrite to formulas in first-order logic. Let S denote a substitution, E an expression, then $[S]E$ denotes the result of applying S to E. The *basic substitution*, denoted $v := E(V)$, where E is an expression on variables V, states that, when the operation completes, the value of variable v is $E(V)$, where the values of the variables appearing in this expression are taken when the operation initiates.

Esterel

3 Tool comparison

In order to verify how the current tools can be modified to reflect the industrial needs, I prepared a brief comparison of some existent tools. I have restricted this comparison to some tools that I know better and that have been used in my application field, that is, railways application. Those tools are, Atelier B, RODIN and SCADE.

3.1 Methodology

The "Oracle" I used to determine the classification of each tool in each category was my personal feeling since a more detailed research was not performed so far, but even so, in the last 3 years I could collect some comments from people I have been training, so I hope it may be helpful.

It's also prudent to state the maturity differences among these tools. While SCADE and AtelierB have been in the market for a long time, RODIN is about to be released in its first oficial version (version 1.0) what means that the first two have already received many feedbacks from its industrial users helping them to change the directions when to users were not satisfied (it was the AtelierB case, where after a lot of complains about the user interface, change completely its GUI), while the last one had no time yet to receive or to implement completely these feedbacks.

The comparison methodology was based on three aspects, as follows:

- *capability:* in this case I try to verify how these tools can satisfy project constraints
- *usability:* basically, what's the difficulty the user faces when trying to use the tool
- *adequacy to the current development process :* I mean, how the tool can better fit in the process without causing too many changes in the way it was performed so far

To make a classification of these aspects I used a simple ranking method, as follows:

- *1* Very dificult
- *2* Medium
- *3* easy

The results are presented in table 1.

3.2 Chart comparison

Aspect	**capability**	**usability**	**adaptation**	***Results***
AtelierB	2	1	2	*5*
RODIN	2	2	1	*5*
SCADE	2	3	3	*8*

Table 1. Comparsion table

To justify those results it's possible say that:

- AtelierB
  - the capability to solve the project constraints is not so bad, but you do need to know a lot of the formal language and constructs to be able to have easy proof obligations.
  - Although, the version 4 of AtelierB supplies a real better usability, all comments I have so far are based on the previous version where the lack of a good User Interface makes its usage painful.
  - Since it allows to go from the specification to the code it can be considered as a good tool for that purpose, but as the interactions during the middle phases (refinements) are some times, painful, it can not receive the higher grade.

- RODIN
  - As it's not so different from AtelierB, similar results could be seen, I mean, the capability to solve the project constraints is not so bad, but you do need to know a lot of the formal language and constructs to be able to have easy proof obligations.
  - the way that RODIN was constructed helps a lot a non experienced person, as you just need to fill down some fields to have a basic specification, but a lack of text editor that could help more experienced person and speed up the specification process lowers its classification
  - As, at the moment of the evaluation, there were no possibilities of decomposition, and the ability to help only in the system specification phase, turn it in a difficult tool to be used in the current process.
- SCADE
  - Even based on a different concept, where formal methods are behind the scene, it has a great capability to deal with project constraints, but you still need some formal background to construct correct models.
  - As it was built from the very beginning to be an industrial tool, its usability is its strongest point, with a good interface and a lot of fancy features that captivate the user. A lot of things can be done based on templates and patters, what helps a lot as well
  - Besides the capability to go from the specification to the code, it has also some other complementary tools that help you in important auxiliary tasks in the project like, requirement management, traceability, etc..

4 Experiences

Basically, my experiences in formal methods are both as a practitioner and as a researcher. In the last 3 years I've been trying to introduce formal methods in the projects I have worked on, and I can say that even if they can not fulfill all industrial needs they can help a lot to better model the development process and the resultant product (or software).

I can summarize 3 different projects that I've been working on recently and what I could achieve so far:

4.1 Signaling system

European companies whom develop signaling systems for railway applications are known as one of few that use formal methods during the development process. It also true that even doing that, their branches around the world do not follow the same concept. During 2008, using B method and the associate tool, AtelierB, I participated in a revalidation process o a signaling system.

As, for this kind of system, there are always a start point, I mean, the new project is, normaly, based on the previous one, the task was to implement new functions and after that revalidate all the system (it's required by the standard, IEC 61508, for systems classified as SIL4, at the moment you change any function, the whole system has to be revalidated).

As the system was previously developed in B, this kind of task became a trivial one. (not only by using B method but also because the related tools, AtelierB in this case, are powerful enough to keep track of the changes and reprove only what is realy needed). Basically, the changes were applied in the abstract model, and after that were reflected in the refinements an implementatio. New prof obligations were genereated and the affected older ones were reapplied.

The result of the complete process were that after the deployment of the system no failures where detected. The associated costs in this development were less than in a traditional process as there were no needs of maintenance changes and the necessary time dedicated to testing was really short. But again, this job was performed in a company that has been using formal methods for a long time.

4.2 Door system

In order to verify the consistency of a door system specification, we used RODIN as a proof of concept, and it was possible to show the benefits of this approach to the final customer. This job was developed based on a small portion of the natural language specification, but we could verify at least 3 contradictions or inconsistencies on it. The objective was to help them to rewrite the specification based on the result of the verification of the formal model.

The natural language specification is more than 100 pages long, and the needed information is spread out over all this specification. As an example of one contradiction we found, take this two statements that were in the specification:

- The train is not allowed to move while the at least one door is open;
- If the emergency buttom is pressed, the respective door must open when the train speed is bellow 6 km/h.

With this simple example is easy to see a contradiction, moreover by the way it's presented here. But those statements were spread out in the specification, so the direct comparation like here were not so clear.

We can see that there is a situation where the door's behavior is in contradiction as the door should not open until the train is completely stopped, but also is demanded that bellow 6 km/h, in a emergency situation it should be open.

the machine in figure 4.2, represents this specification, and the PO in figure 2 represent the contradiction.

MACHINE Open_contradiction
VARIABLES
`train_stoped` boolean. when the train is stoped it's value is TRUE
`train_low_speed` boolean. when the train speed is below 6km/h it value is TRUE
`door_authorization` boolean. when the train is allowed to open doors it's value is TRUE
`emergency_buttom` boolean. if the buttom is pressed, it's value is TRUE

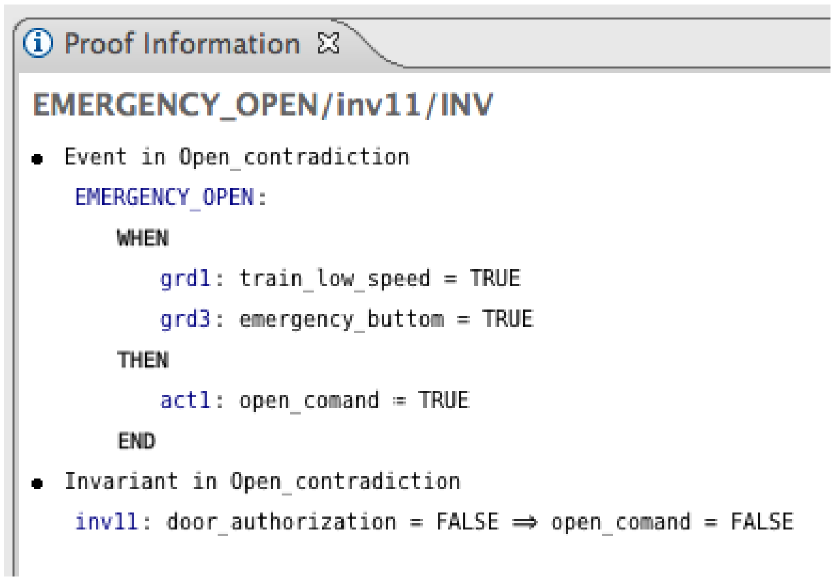
`open_comand`	boolean. if true, command the opening
`train_speed`	NAT. real speed

INVARIANTS

```
    inv1 :  $train\_stoped \in BOOL$
    inv2 :  $door\_authorization \in BOOL$
    inv3 :  $train\_low\_speed \in BOOL$
    inv4 :  $emergency\_buttom \in BOOL$
    inv5 :  $train\_stoped = TRUE \Rightarrow door\_authorization = TRUE$
    inv6 :  $train\_stoped = FALSE \Rightarrow door\_authorization = FALSE$
    inv7 :  $train\_stoped = TRUE \Rightarrow train\_low\_speed = TRUE$
    inv9 :  $open\_comand \in BOOL$
    inv10 : $train\_speed \in \mathbb{N}$
    inv11 : $door\_authorization = FALSE \Rightarrow open\_comand = FALSE$
EVENTS
Initialisation
    begin
        act1 : $door\_authorization := TRUE$
        act2 : $train\_stoped := TRUE$
        act3 : $train\_low\_speed := TRUE$
        act4 : $emergency\_buttom := FALSE$
        act5 : $open\_comand := FALSE$
        act6 : $train\_speed := 0$
    end
Event  $EMERGENCY\_OPEN \widehat{=}$
    when
        grd1 : $train\_low\_speed = TRUE$
        grd3 : $emergency\_buttom = TRUE$
    then
        act1 : $open\_comand := TRUE$
    end
Event  $LOW\_SPEED\_MONITOR \widehat{=}$
    when
        grd1 : $train\_speed \leq 6$
    then
        act1 : $train\_low\_speed := TRUE$
    end
Event  $ZERO\_SPEED\_MONITOR \widehat{=}$
    when
        grd1 : $train\_speed = 0$
    then
        act1 : $train\_stoped := TRUE$
        act2 : $train\_low\_speed := TRUE$
        act3 : $door\_authorization := TRUE$
    end
Event  $AUTHORIZARION\_RELEASE \widehat{=}$
    when
```

```
        grd1 : train_speed > 0
    then
        act1 : door_authorization := FALSE
        act2 : train_stoped := FALSE
        act3 : open_comand := FALSE
    end
Event  LOW_SPEED_RELEASE ≙
    when
        grd1 : train_speed > 6
    then
        act1 : train_low_speed := FALSE
        act2 : train_stoped := FALSE
        act3 : door_authorization := FALSE
        act4 : open_comand := FALSE
    end
END
```

It's clear that to discharge this PO, figure 2 is not a question of correct the model, but the natural language specification must be changed to avoid this kind of ambiguities or contradicitions.



```
EMERGENCY_OPEN/inv11/INV

• Event in Open_contradiction
  EMERGENCY_OPEN:
     WHEN
        grd1: train_low_speed = TRUE
        grd3: emergency_buttom = TRUE
     THEN
        act1: open_comand ≔ TRUE
     END
• Invariant in Open_contradiction
  inv11: door_authorization = FALSE ⇒ open_comand = FALSE
```

Fig. 2. PO to be discharged

In this case three different approaches or options were proposed, as follows:

1. The train is not allowed to move while the at least one door is open, *unless in a emergency situation*;

2. The train is not allowed to move *over 6 km/h* while the at least one door is open;
3. If the emergency buttom is pressed, the respective door must open when the train *stops*

The first option was choosen by the custumer, and the specification and model was changed to reflect this new constraint.

Based on this simple example, it was easy to present the formal method benefits where the imposibility to introduce ambiguities and contradictions was stated.

The objective now is to try to represent the complete specification of one train sub-system (probably the door system) in Event-B[16], and reformulate the natural language specification in a better representation. At least, pointing out the itens that needs to be revised to create a more consistent specification.

4.3 Platform screen doors

Platform Screen Doors, aka PSD, is a door system that is installed in the platform stations to avoid people to fall down to the track. The safety related issues are even higher than train door system as, people get used with it, and a dagerous situation can lead to severe accidents[11]. For example, if the train departure with doors in PSD open, the train can easly hit someone.

This kind of system is been installed in Metro São Paulo, Brazil, and a company from Korea was hired to develop and install the system. The same standards must be applied in order to guarantee that the desired safety level (in this case SIL 3[17]) will be met.

Besides the safety constraints (that by itself its a huge problem) there is no room to rework as the whole system needs to be in operation at the end of 2009. As the first phases of the "V model", take a lot of time, while in a formal process, it will be no time, after all, for a huge amount of tests.

As support standard, the IEC 62279[18] can be used to guide on the necessecity of documentation that should be generated to prove that the needed care was taken during the development process. The documentation that needs to be generated is as follow:

1. System Requirements Specification
2. System Safety Requirements Specification
3. System Architecture Description
4. System Safety Plan
5. Sw Configuration Management Plan
6. Sw Verification Plan
7. Sw Integration Test Plan
8. Sw/Hw Integration Test Plan
9. Sw Requirements Specification
10. Sw Requirements Verification Report
11. Sw Architecture Specification

12. Sw Design Specification
13. Sw Arch. and Design Verification Report
14. Sw Module Design Specification
15. Sw Module Test Specification
16. Sw Module Verification Report
17. Sw Source Code Verification Report
18. Sw Module Test Report
19. Sw Integration Test Report
20. Sw/Hw Integration Test Report
21. Sw Validation Report

Moreover, it's requested that from the detailed specification to the unit tests a formal method should be used, as said also in section 2.

Another problem that was faced is the lack of knowledge about formal methods and development process by the team in charge of the project. As meintioned before, the culture of a structured process is not planted, at leaast in my experience, in Korea.

Based on all of these considerations, SCADE tool was selected to help on these tasks. As meintioned in section 3, SCADE seems to be the best choice for non-fromal method people.

From the list presented before, it's possible to say that the itens from 10 to 19 can be performed, or automaticaly or is supported by the tool. All the formalism is performed behinf the scenes, so the user can feel confortable in develop what is realy needed from his point of view.

Even with this simplistic view, it was possible to verify that the formalism, and moreover, the capability to model checking and theorem proving, helped to better the quality and consistence of the generated documentation and to verify missing points and inconsistences.

As an example, let's take two functions that should be modeled, based on the first requirement specification of one of the PSD system equipments.

The equipment is called, PCM, and it is in charge of control the door open and close functions while in manual mode, what means, while PCM is enabled.

The two extractions from the Software Requirement Specification are as follows:

– *open command* If PCM is enabled, and the OPEN buttom is pressed longer than 1 second, the OPEN command has to be generated.
– *close command* If PCM is enabled, and the CLOSE buttom is pressed longer than 1 second, the CLOSE command has to be generated.

Using SCADE, it was modeled like figure 3

One more time, it's easy to realize that there are a lot of missing information and contradictions. For example, it's not possible to say if it's correct or not, based only in the specification, to generate an close command while the open command is present, and vice versa. There are no information that could be used to determine when the command (doesn't matter open or close) should be turned off. One can figure out a lot of different needs, this is not the objective.

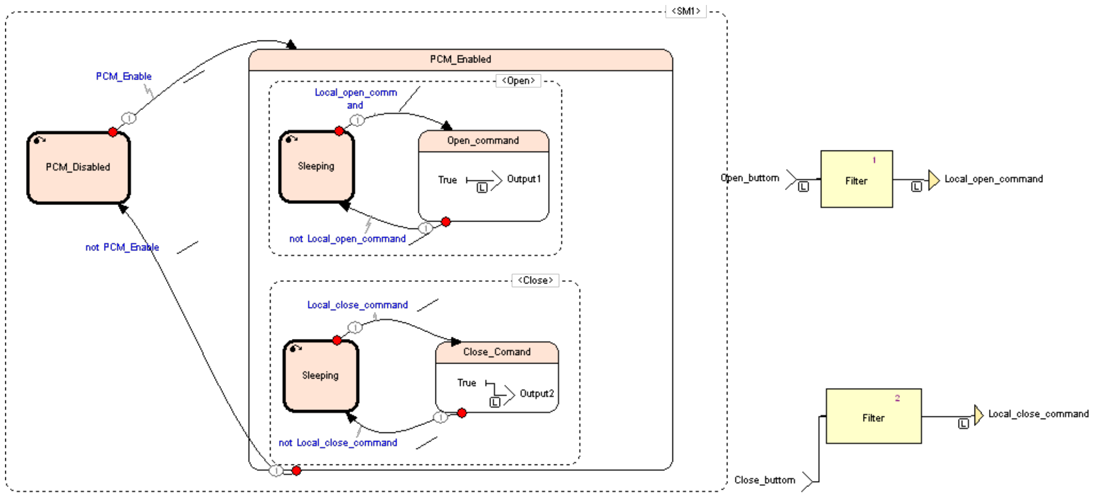


Fig. 3. SCADE model

The main objective here was to present that a simple way to formalize the development process, whether or not, with heavy formal methods, helps a lot to find these kind of problems.

It's an ongoing project, and I hope to present some strong evidences to support these assumptions.

4.4 considerations

Some other points I'd like to point out. Despite all odds, and as pointed in [19], it was not necessary to have someone with strong knowledge in mathematics, altough the basic concepts were needed. Moreover, it was not necessary a big team in none of the described projects in order to sucefully cary on the project. In the second project I cited before, just one person did all the work.

In all of these projects, the most dificult task, and the one that took more time was the requirement elicitation and analysis. Even if it not direct related to formal methods, as it has to be carried on does matter the process you adopt, the goal to build a formal model helps during the classification and elaboration of each requirement forcing them to be complete and non ambiguous.

At the end, the time (and money) that is spent in the earlier phases of the development process is greater than in a normal development, but the time (and much money) that is spent in tests and rework are definitely less. In the case of the second example (Door system), even using the formal methodology only as a support tool, the resulting test cases were much more effective, and the period of tests was shortened by 2 months (from 6 months to 4 months).

Based on these experiences, and others, I summarize in section 5 some features that I think could be included in RODIN platform.

5 Gaps or needs

In this section it is summarized some of expectations about the future of supporting tools and point out some characteristics that is presumed as necessary. Most of them are being prepared, but even though some key points for each one might be pointed out.

- *Requirements* - It's a fact that requirement problems are responsible for more than 40% of the total problems in a project 4. With this in mind, this is the most important feature that should be integrated in RODIN platform. From my point of view there are some characteristics that might be useful:

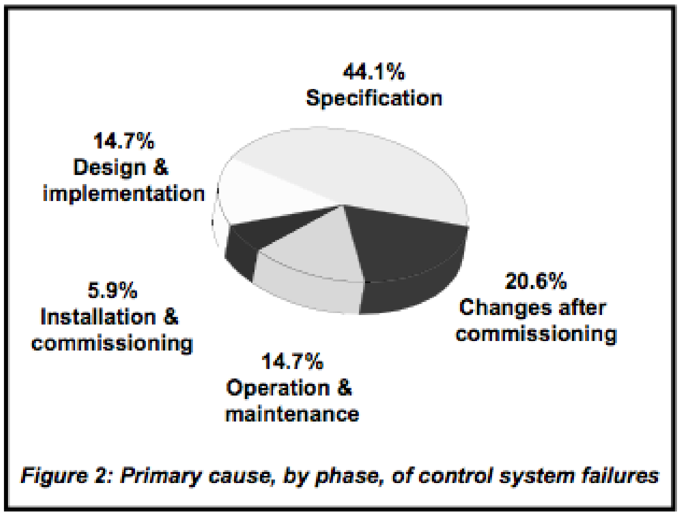


Fig. 4. Requirement problems from [17]

  - It would be great if we could develop a "natural language dictionary", that would be used to rewrite the specification in a way it could be better understood.
  - Based on this redefined specification it would be good if we could directly "convert" this specification in our abstract model, avoiding with that the insertion of human errors
  - It would also be good if we could create a tool that, based on the formal model, could write back a natural language specification. This is crucial when we model our model manually and after all we need to present it to the customer for its approval.
  - to help traceability, it would be good to develop, more than a simple "natural language dictionary", a methodology to annotate the requirement file, and with that we would be capable to verify the coverage of this requirement.

– *Traceability* - Even if it's related also to requirements, I think RODIN platform might have also capability to:
  - to be able to track forwards, I mean, if we change something in the abstract model, it would be good if RODIN platform could point out the possible refinements that should be verified in order to meet the changes.
  - in the same way, it should be able to track backwards, and point where we should verify if we made some changes (intentional or not) in our refinement machines.
  - more crucial, it would be good if with RODIN platform we could be able to track back and forward all the requirements and any changes could be highlighted. Moreover, with this ability, we would be able also to verify if all requirements were fulfilled or not.

– *intermadiate languages* - This is something that's already been done by UMLB plugin, but I think that one interesting feature is missing. Besides the ability to create state machines, for example, the ability to execute these models would be gratefully appreciated. With that, we would be able to verify if our assumptions are correct, with no need to go inside the proof obligations.

– *test case generation* - This, in my opinion, is one of the biggest gaps in industry right now. All generated tests are based on specialist feelings, and usually, what is tested is not exactly what should be. As a result, after a long time testing the system, at the moment it's put in operation some failure happens, and the test generation phase has to begin again in order to address that specific failure. This routine happens several times until the product can be finally released.
  I have strong feelings that the Proof Obligations are the basic source to generate test cases that are necessary and sufficient. If those proofs are necessary and sufficient to validate the specification, why not use those proofs to generate the test case scenarios?

6 Conclusion

As it's more a positional paper than a research paper itself, I will present my personal conclusions to you. The application of formal methods in industry is growing, however most of the times as a result of some projects involving academia and industry, like DEPLOY project.

It's clear that outside Europe, formal methods usage is still incipient, and more effort in showing the benefits of that use is needed. In order to facilitate this approach we need tools that do not scare the customer in a first sight, otherwise the fear not to perform a good job will be always greater than the possibility of creating better products.

If these barriers could be broken, I think that the use of formal methods would spread out really fast.

If the introduction of the features I mentioned before could be a reality, it would be a great step for this project.

If the managers are open mind, and admit waiting a bit more in the begining of the development to see real results, (light or heavy) formal methods application could be a lot cost-effective and can, at the end, decrese the costs of the whole project by decreasing the costs in test and maintenance phases.

7 Aknowledgements

I'd like to thank to all univiersities and companies cited in this paper for their support providing me recomendations and tools.

Finally I'd like to thank to Mr. Hrvoje Belani, who devoted his time to review this paper and mostly for his comments which help me to include substantial points in this work.

References

1. ClearSy: Atelierb 1
2. Butler, M., Hallerstede, S.: The rodin formal modelling tool. deploy-eprints.ecs.soton.ac.uk 1
3. Esterel: Getting started with scade. (Sep 2007) 1–148 1, 5
4. Bowen, J.P., Stavridou, V.: The industrial take-up of formal methods in safety-critical and other areas: A perspective. In: FME '93: Industrial-Strength Formal Methods, First International Symposium of Formal Methods Europe. Volume 670 of Lecture Notes in Computer Science., Odense, Denmark, Springer (1993) 183–195 2
5. Commission, I.E.: IEC 61508 - Functional safety of electrical/electronic/programmable electronic safety-related systems. International Electrotechnical Commission Standards (1998) 2, 3, 5
6. CENELEC: Software for Railways Control and Protection Systems. EN 50128. (1995) 2, 5
7. Abrial, J.: The b book: Assigning programs to meanings. books.google.com (Jan 1996) 2, 5
8. Hwang, J., Jo, H., Jeong, R.: Analysis of safety properties for vital system communication protocol. Electrical Machines and Systems, 2007. ICEMS. International Conference on (2007) 1767–1771 3
9. Abrial, J.: Formal methods: Theory becoming practice. Journal of Universal Computer Science (Jan 2007) 5
10. Jetley, R., Iyer, S., Jones, P.: A formal methods approach to medical device review. COMPUTER (Jan 2006) 5
11. Lecomte, T., Servat, T., Pouzancre, G.: Formal methods in safety-critical railway systems. Proc. Brazilian Symposium on Formal Methods: SMBF (Jan 2007) 5, 12
12. RTCA, I.: DO-178B, Software Considerations in Airborne Systems and Equipment Certification. (1992) 5
13. Squair, M.: Issues in the application of software safety standards. Proceedings of the 10th Australian workshop on Safety critical systems and software-Volume 55 (2006) 13–26 5
14. Bernardeschi, C., Fantechi, A., Gnesi, S., Larosa, S.: A formal verification environment for railway signaling system design. Formal Methods in System Design (Jan 1998) 5

15. Boussinot, F., Simone, R.D., ENSMP-CMA, V.: The esterel language. Proceedings of the IEEE (Jan 1991) 5
16. Metayer, C., Voisin, L.: The event-b mathematical language 12
17. Bell, R.: Introduction to iec 61508. Proceedings of the 10th Australian workshop on Safety . . . (Jan 2006) 12, 15
18. Commission, I.E.: IEC 62279 Railway Applications Communications, Signalling and Processing Systems Software for Railway Control and Protection Systems. International Electrotechnical Commission Standards (2002) 12
19. Bowen, J., Hinchey, M.: Ten commandments of formal methods ten years later. COMPUTER (Jan 2006) 14