DELIVERABLE D1.2 – WORK PACKAGE 1

PROOF OF CONCEPT APPLICATION IN RAILWAY DOMAIN

ADVANCE

Grant Agreement: 287563
Date: *March 21st 2013*
Author: *Fernando Mejia, Alstom; Damien Ledoux, Systerel*
Status: *Final version*
Reviewer: *Michael Butler, University of Southampton*
Reference: *D1.2*
Issue: *2*

Partners / Clients:

FP7 Framework Programme ***European Union***

Consortium Members:

University of Southampton ***Critical Software Technologies*** ***Alstom Transport*** ***Systerel*** ***Heinrich Heine Universität Düsseldorf***

Contents

1. Introduction

This deliverable reports on the proof of concept application of ADVANCE methods and tools to the railway domain case study. The objective of the proof of concept phase of WP1 was twofold:

- to assess the suitability of Event-B for modelling and verifying of system-level safety properties of a generic railway interlocking system;
- to assess the existing ADVANCE tools for their suitability and to identify gaps in the existing tools.

In summary, as elaborated in this deliverable, our conclusion from the proof of concept phase is that Event-B is suited to modelling and verification of system-level safety properties of a generic railway interlocking system. The existing Rodin-based ADVANCE tools provide good support for modelling and verification of system-level safety properties but, through the proof of concept work, we identified some gaps in the tool chain leading. These tooling requirements are outlined and are being addressed by WP3-WP5 of ADVANCE.

The full development of the railway domain case study has been decomposed in five tasks:

1. Requirements definition
2. Modelling and animation
3. Refinement plan
4. Safety analysis
5. Transition from Event-B to B

In the proof of concept phase we focused on the first two steps but also draw some initial observations about the remaining tasks.

The deliverable is organised as follows. The second section of the deliverable presents briefly the object of the case study, the interlocking dynamic controller. The remaining sections of the deliverable present the previously mentioned tasks: its purpose, its inputs and outputs, the relevant ADVANCE methods and tools applied to achieve it and, for those tasks that have been carried out, the feedback of the application of those methods and tools.

2. Interlocking Dynamic Controller

Although interlocking systems have been designed to meet the system safety requirements (i.e. no collision of trains, no injury of passengers or maintenance staff), it is very difficult, if not impossible in many cases, to prove formally that they effectively do it because this results from the execution of independent basic actions whose effect on the system is hard to formalise.

To overcome this situation we propose to create a new interlocking component, less complex than the interlocking system itself, whose purpose is explicitly to ensure that safety requirements are met. This new component, called an interlocking dynamic controller, will ensure that the interlocking system manages safely the critical trackside signalling equipment (e.g. points, signals, emergency stop plunger, etc.) according to the current state of trackside equipment, the movement of trains and the supervision system's commands. In other words, the dynamic controller system will check that the interlocking system commands do not create a dangerous situation in which trains may collide or injure passengers or maintenance staff.

3. Requirements Definition

As said above, our intention is to develop a dynamic controller of an interlocking guaranteeing explicitly the system's safety requirements regardless of the internal behaviour of the interlocking system. We shall thus analyse the behaviour of the dynamic controller at the system level, interacting with an interlocking system seen as a black box and with an "environment", that is to say the rest of the signalling system: trains, points, signals, supervision system and so on.

In this context, requirements definition consists in expressing and allocating the requirements to be met by the dynamic controller and the requirements on the interlocking system and on the environment on which the dynamic controller relies.

The inputs of this task are the signalling system specification, the signalling system requirements and the interlocking system specification.

The output of this task is the requirements document for the interlocking dynamic controller. This document will also include the simplifying assumptions for the case study.

3.1 ADVANCE Concepts or Tools Applied

Requirements will be the reference for modelling the properties of data and events of the Event-B models of the overall signalling system. Therefore, requirements must be precise, concise and expressed at the right level of abstraction. With this purpose in mind we shall define precisely a reduced set of concepts with which we shall express requirements rigorously.

Requirements will neither be arranged nor structured in the requirements document. This will be done later, in the subsequent tasks.

We shall represent the allocation of a requirement to a particular part of the system by a unique tag next to the text of the requirement. For instance the requirement below is the first requirement allocated to the environment:

A track network may contain some track circuits, points and signals.	ENV-1

Requirements and requirements traceability will be managed with the ProR tool developed by the University of Düsseldorf [2].

3.2 Feedback on Requirements Definition

As said before, the purpose of this task is to provide the various requirements to be met by the components of the system. For this, we must express our needs and its associated constraints.

It is difficult to write such a document because we do not know what are the concepts that we shall have to handle and the constraints that we shall have to express. The aim is to have the least concepts and constraints in our specification. Indeed, having the fewest concepts possible while still representing the safety requirements adequately is important because the concepts will eventually be used in our Event-B model. Moreover, if the model is overly complicated, verification will also be difficult. Finally, having the fewest constraints possible is very important; all these constraints must be validated, and it is very easy to introduce excessive constraints that do not comply with the actual

world. For example, in our context, we can introduce a constraint which indicates a train does not pass a red signal, but this is false because a train may pass a red signal if it has not the necessary distance to stop in front of the signal. This is explained in more detail in the document found in the appendix.

Before writing this document, we have decided to make a preliminary Event-B model to identify the different concepts that we shall have to handle (cf. section §4). More specifically, we have taken the document describing the hazard tree established by Alstom. This document contains various hazard events of system.

Next, we have selected the main safety properties which our system must meet. These properties are:

- Trains must not derail
- Trains must not collide into each other

From these properties, we have tried to express them formally. For that, we had to introduce the concept of train, but a train circulates on a path that is composed of points. Moreover, a train is allowed to move according to the signal aspect. All these concepts once introduced have begun to establish a preliminary Event-B model. Naturally, we had to introduce constraints on the system, for example a train cannot split while on track, a train cannot enter or leave in the middle of the track. Furthermore, the proof and the animation have allowed us to improve definitions and refine our constraints.

With this preliminary model, we had a better idea of the concepts and constraints that we shall manipulate and therefore to introduce into our specification.

When writing the requirement document, we have decided to use the same format as that defined by the example "train system" in [1]. Indeed, we have described our system and our needs and we have emphasized the various requirements expressed. This manner allows a quick reading of the various requirements. Moreover, the writing of this document, has led us to reflect on the perimeter that we would give to our case study. Indeed, we must be careful not to add features that would be useless or not bringing new challenges to existing functionality.

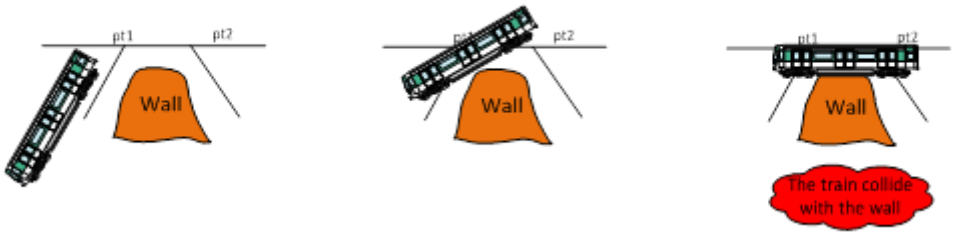
The first draft of the requirement document manipulated trains with and without driver and all the complexity it brings. After some consideration, we have decided to treat only driverless trains, because these are the trains that will be most common in the future. Indeed, the trains with drivers bring complexity in the model that should not be present at this time. The aim of our case study is to test the feasibility of our approach. We must be limited to essential features and see if they can already satisfy our case study. By considering only driverless trains, this allowed us to simplify the management of signals and the signal approach locking areas. Indeed, this information is different than it is for a train with or without driver.

Moreover, we wanted to check that the points were correctly positioned according to the routes defined by the technical plan. But, we did not want to handle the route concept in the requirement document. So, we identified a property that seemed interesting to treat and will force us to check the point position. Indeed, we wanted to check that no collision between a train and a system structure occurs. Specifically, a train can collide with the railway tunnel wall when the points are not

correctly positioned, as illustrated in the figure below. In this example, there are two points. These points must respect the following positions to prevent an accident:

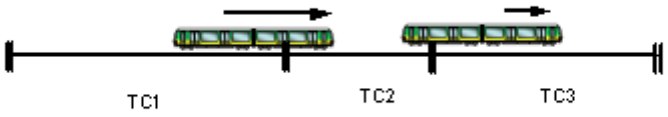
- Point “pt1” to the right and point “pt2” to the left
- Point “pt1” to the left and point “pt2” to the left
- Point “pt1” to the right and point “pt2” to the right

In our case, the point “pt1” is positioned to the left and the point “pt2” is positioned to the right. If the points are close and the train car is long enough then the train can collide with the railway tunnel wall.

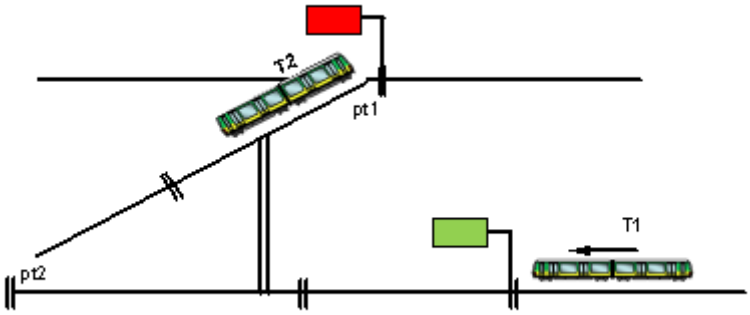


This property has been discarded because it is not considered essential, and it is not the responsibility of the interlocking. Indeed, this problem should be checked when building tracks and installing rails. This means that a track layout like this should never be built.

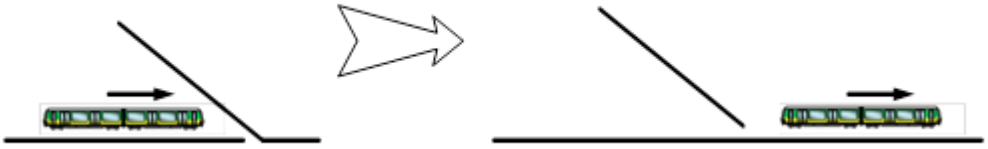
Similarly, the property of the rear-end collision has not been retained in our proof of concept case study. Indeed, we assume that a rear-end collision does not occur, as illustrated in the figure below. We have decided to treat only driverless trains, so in this case rear-end collisions are prevented automatically by another subsystem (automatic train control). This risk does not exist in our context.



Nevertheless, one property has been added, which concerns trap points. This property comes from Alstom‘s hazard analysis. This property is considered of second order since it avoids a collision if a train inadvertently passed a signal leading to the mainline, as illustrated in the figure below. Indeed, in our example, the train T2 passed the red signal, and it is stopped. If the train T1 moves forward, there will be a fouling collision. A fouling collision takes place when tracks are so close physically that if there was a train on both, then a collision would occur (indicated by the double lines). Therefore, if the point "pt1" was positioned to the right, the train T2 would not be in the fouling area and no risk would be present.



Furthermore, consideration of trailable points has not been retained. Indeed, these points can change position during the passage of a train, as illustrated in the figure below. In this example, the point is right before train passage, and then train changes the position of the point to the left.



This type of point is not in accordance with the requirement which indicates that a train cannot move a point. We have decided not to consider these points because the test track does not have these points. Indeed, before starting our case study, Alstom has chosen a test track "*Parc des Expositions*" in Paris. This track will allow us to test our approach and verify its feasibility on a concrete example.

Finally, during our rereading, we have also identified features that could complicate the model. For these, we have decided to introduce them later. These features are:

- The stabling tracks. They allow trains to park after the service for example. The difficulty is when a train has to resume service. It is useful to treat this feature because it will allow us to observe the transition from a degraded to a nominal mode.
- Cancellations areas and cancellations of point position control. They can unlock the system when a sensor is defective. Indeed, safety ensures that if a track circuit sensor fails the track circuit is seen as occupied or when a point position sensor fails the point is not seen as positioned. In these cases, the system can deadlock. Therefore, cancellations have been added to bypass sensor failures. They involve agents who ensure that no-risk is possible.

As we mentioned above, requirements will be represented in the B model with varying degrees of ease. Indeed, a requirement could be expressed directly by an invariant, by an event guard, by an axiom, by many events, etc... It is important to check that all requirements defined in the document are accurately defined in the model. For this, we must establish traceability between the model and requirements document. To facilitate this, it would be interesting to use ProR [2], which seems to help this activity.

It is important to note that the requirements document was written quite easily because Alstom and Systerel have significant experience in railway signaling and especially on interlocking. Therefore, our

experiments allowed us to directly manipulate the most important concepts. Nevertheless, each system has these features, so we lean on the principles defined by the interlocking product.

Writing such a document is essential. Indeed, it helps to clarify ideas and to delimit the system perimeter. Moreover, we know beforehand that the requirement document will be changed because some requirements must be added or the description of some concepts refined. Finally, we were careful not to over-constrain the system by not explaining how we would realize our dynamic controller. The purpose was not to give a solution but to express our need.

4. Modelling and Animation

The development of a preliminary model, in parallel with the requirements definition task, will help us to identify the appropriate concepts for the definition of requirements and to find best ways to model the dynamic controller. Moreover, it will allow us also to manipulate Event-B and the tools promoted by ADVANCE: Rodin and its plug-ins that we shall use to create and prove the preliminary Event-B model and ProB that we shall use to analyse the dynamic behaviour of the model.

The inputs of the task are the requirements documents and the refinement plan.

The outputs of the task are the Event-B models of the signalling system and the feedback on this task.

4.1 ADVANCE Concepts or Tools Applied

Relevance, provability, readability, maintainability and traceability are the criteria that determine the quality of a formal model.

A model is *relevant* if its behaviour meets the user needs in all operational situations. Relevance is obtained notably by the possibility of analysing statically and dynamically the behaviour of the model in different operational situations.

A model is *provable* if it can be proved formally and completely at a reasonable cost. Provability depends very much on the concepts chosen to describe the system and on the simplicity of their formalisation. Relevance and provability ensure the appropriateness and the correctness of the model regarding the requirements of the system.

Readability facilitates the understanding and the analysis of the model by peers. Like provability, readability depends on the concepts chosen to describe the system and on the simplicity of their formalisation.

Maintainability reduces as much as possible the impact of the modification of one part of the model on the other parts of the model. Modularity of the model is critical for its maintainability.

Traceability means that every feature of the system can be related to pieces of the model and that every piece of the model can be related with a feature of the system. At the first step of development traceability depends on the concordance between the concepts used to describe the system informally and the concepts used to describe the system formally. At later steps of development, traceability is ensured by formal refinement.

4.2 Feedback on Modelling and Animation

4.2.1 Event-B Modelling

As said in the previous section, we have developed a preliminary model to put us in context and to manipulate ADVANCE tools (ie. Rodin, its plug-ins and ProB). This phase has been important because it allowed us to see the main concepts that we had to handle in our model. This was very helpful when writing the requirements document. Indeed, it helped us to avoid introducing unnecessary concepts.

Initially, we were inspired by the example "train system" in [1]. In particular, the approach to separate the front and rear of the train during his progress is interesting because it helps to have a model with an indefinite train length and to simplify the train movement.

Moreover, this model has allowed us to introduce a set of hypotheses which were reused in the requirements document.

To finish, we have also expressed safety properties in the model and we realized they were complicated to read and prove with the Rodin tool, as illustrated by the property below. This property checks that the points are correctly positioned under and ahead of the train to the next signal.

```
∀tr·(tr∈ dom(v_trains) ⇒
        (∀tr_p·(tr_p ∈ t_block×t_traffic_direction ∧
                tr_p ∈ v_trains(tr) ⇒
                (∃nn,track·(nn ∈ ℕ1 ∧
                 track ∈ ((1 ‥ nn) → (t_block × t_traffic_direction)) ∧
                 track ≠ ∅ ∧
                 track(1) = tr_p ∧
                 (∀ii·(ii ∈ 1‥card(dom(track))−1 ⇒ (track(ii) ∈ dom(v_next_block)))) ∧
                 (∀ii·(ii ∈ 1‥card(dom(track))−1 ⇒ (v_next_block(track(ii)) = track(ii+1)))) ∧
                 (track(nn) ∈ dom(v_next_block) ⇒
                       (track(nn)↦v_next_block(track(nn))) ∈ ran((c_upstream_block_of_signal⊗
c_downstream_block_of_signal))) ∧
                 (track(nn) ∈ dom(v_next_block) ⇒
                       (v_trains_status(tr)((c_upstream_block_of_signal⊗
                                             c_downstream_block_of_signal)∼
                                            (track(nn)↦v_next_block(track(nn))))=c_train_stop))
∧
                 (track(nn) ∉ dom(v_next_block) ⇒ track(nn) ∈ c_extremity_blocks)
                )))))
```

We have decided to use the Theory plug-in of Rodin to introduce new mathematical operators allowing us to simplify the expression of this property and its proof.

For modelling, we were realized that a train corresponded to a chain moving on a track network. So we decided to create new mathematical operators to manipulate chains on a graph. Indeed, a track network can be represented by a graph. For this, we have used the definition of a chain proposed in the chapter "Finite Lists" from [1]. Moreover, we have defined several operators to manipulate these chains and thus to move the trains.

The main improvement of the model has been to use the Theory plug-in. Indeed, it allows us to express the operators, theorems, rules, allowing us to simplify the model and proof.

We have decided to model trains by chains and we have introduced operations to manipulate these chains. They allow us to move forward or backward a train, making a turnaround. These operations were thereafter used in the model, making it much more readable. Moreover, rules have been added for performing the proof of our model. They allow us to guarantee that each operation maintains the chain characteristics. This means that a train is always a train.

These new operators have allowed us to express our properties differently. Indeed, now we check that all trains correspond to a chain on dynamic network, as illustrated by the property below. If this is true, this means that the points are correctly positioned under the train. This property replaces the previous one.

```
v_trains ∈ t_train ⇸ chain_on_graph(v_next_block)
```

During the rewriting of the model with the new operators we have abstracted concepts to try to keep handling of them to a minimum. The aim is to have the simplest possible model. For example, we merged the risk of fouling collision and face-to-face collision in the same property.

Initially, we defined the risk of face-to-face with the following property. We check that two different trains do not intersect. This means in our case, that there is no collision between two trains. Note that the rear-end collision can also be verified by this property. However, we are in a context with driverless trains. Therefore, the construction of our model is that there will be never a rear-end collision.

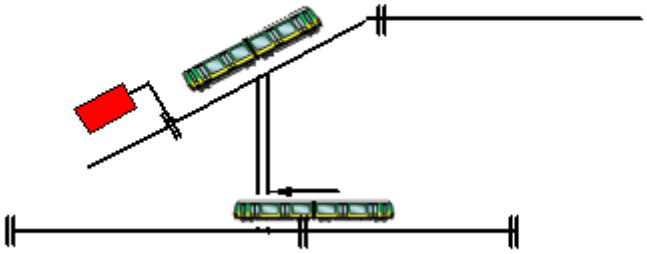
```
∀tr1, tr2· (tr1 ∈ dom(v_trains) ∧ tr2 ∈ dom(v_trains) \ {tr1}
            ⇒
             (v_trains(tr1) ∩ v_rains(tr2))  =  ∅)
```

This property ensures against the risk of face to face, like illustrated by the figure below:

Furthermore, another property modelling the risk of fouling collision had been defined as below. We check that two different trains do not intersect in a fooling area.

```
∀tr1, tr2· (tr1 ∈ dom(v_trains) ∧ tr2 ∈ dom(v_trains) \ {tr1}
            ⇒
             ((dom(v_trains(tr1))×dom(v_trains(tr2))) ∩ c_ouling_blocks)  =  ∅)
```

This property is illustrated by the figure below :



These two properties have been merged into a single property that corresponds to the incompatibility between the blocks, as defined by the property below. We check that two different trains do not intersect in an incompatible area. This area includes the fooling area and the face to face area.

$$\forall tr1, tr2 \cdot (tr1 \in dom(v\_trains) \land tr2 \in dom(v\_trains) \setminus \{tr1\} \Rightarrow ((get\_blocks(v\_trains(tr1)) \times get\_blocks(v\_trains(tr2))) \cap c\_ncompatible\_blocks) = \emptyset)$$

As can be seen, the Theory plug-in has allowed us to simplify the modelling by introducing new operators. The advantage of using this plug-in is that we are able to express rules or theorems on these new operators which will be applied during the proofs. Therefore, the proof will be simplified. While we realize that we have several times the same goal, it is interesting to add a rule that we shall prove independently and apply it thereafter to discharge the goal. A rule is defined by hypotheses and by a goal. These rules handle meta-variables that will be replaced automatically during the proof according to context.

Nevertheless, we have been faced with two main problems.

The first concerns the management of new operators defined in a theory by the Atelier B [4] proof plug-in for Rodin. Indeed, these engines are considered as the most effective and can discharge a large number of proof obligations. When sending the proof obligation to the engine, the hypotheses involving new operators will be translated by "TRUE" and if the goal involves a new operator, it will be translated by "FALSE". This translation may result in that proof obligation not being proved. Therefore, the plug-in did not simplify the level of automation in the proof activity. It is important to note that work is underway to improve this aspect. We shall illustrate this problem by the example below:

We have a proof obligation that manipulates a new operator “Operation”.

$$Operation(F) \land F \subseteq G \Rightarrow Operation(G)$$

This proof obligation will be translated before being sent to the proof engine. In our example, our proof obligation is translated as follows:

$$\begin{array}{|l|}\hline \text{TRUE} \wedge \\ F \subseteq G \\ \Rightarrow \\ \text{FALSE} \\ \hline \end{array}$$

As you can see this proof obligation becomes false. Therefore, this proof obligation will not be proven.

The second problem concerns the rules application. Indeed, when we express a rule, we handle meta-variables that will be replaced during the rule application according to context. These meta-variables are replaced either with the goal or with the hypotheses. Therefore, if a meta-variable is handled only in the hypotheses, this rule cannot be applied in backward mode. We shall illustrate this problem by the example below.

We have written the following rule:

$$\begin{array}{|l|}\hline F \subseteq G \\ \Rightarrow \\ \text{ran}(a \lhd f) \subseteq G \\ \hline \end{array}$$

During the proof, we have the following context, and we want to apply the above rule.

$$\begin{array}{|l|}\hline E \subseteq \mathbb{Z} \\ F = 1 \mathinner{..} x \\ f \in E \pfun F \\ n \in \mathbb{N} \\ G = 1 \mathinner{..} y \\ x \leq y \\ \vdash \\ \text{ran}((1..n) \lhd f) \subseteq G \\ \hline \end{array}$$

We note that the first hypothesis and the goal of the rule are in the proof context. Therefore, all meta-variables can be instantiated. Moreover, the second hypothesis is not present in the proof context, it must be proven. For this, a new sub goal must be generated automatically. Currently, the rule application does not correctly instantiate meta-variables. Nevertheless, work is underway to improve this aspect. With this feature, the rules will be more powerful and proofs will be easier.

This preliminary model is composed of a context, a machine and a theory. The machine includes 14 events, the context includes 25 axioms and the theory includes 11 operations, 5 theorems and 7 rules. This model generates 113 proof obligations, 29 of which are discharged automatically. This low rate of automatic proof is related to integration problems of Theory plug-in and with the Atelier B proof engines.

This model has allowed us to test the Theory plug-in and see its possibilities. Moreover, we have begun to create new mathematical operators that we shall use during the modeling. Furthermore, we have begun to identify the appropriate concepts we shall use in our next model. Finally, it has been very helpful to identify the various requirements.

4.2.2 Animation with ProB

As said in the section §4.1, to obtain a relevant model it is important to check that its dynamic behaviour meets the expected needs. ProB [3] supports doing that check by animation of Event-B and B models. So we tried to animate with ProB the Event-B model presented above but we failed because this tool does not support yet the Theory facility that authorises to define and instantiate generic sets and generic operators on these sets. Evolutions of ProB are underway to support the Theory facility.

Thus we rewrote the Event-B model into a classical B model and generate data representing the RER B's "*Parc des Expositions*" station near Paris in order to animate this model with ProB. The animation of the model disclosed that two axioms were not well defined. However, these two axioms do not have a significant impact on the proof.

We improved this first model and produce a more relevant, realistic, one. The new model distinguishes the dynamic controller, the interlocking system and the environment (cf. section §3) and takes into account possible failures of trackside equipment and asynchrony between the actual state of trackside equipment and the state of the dynamic controller.

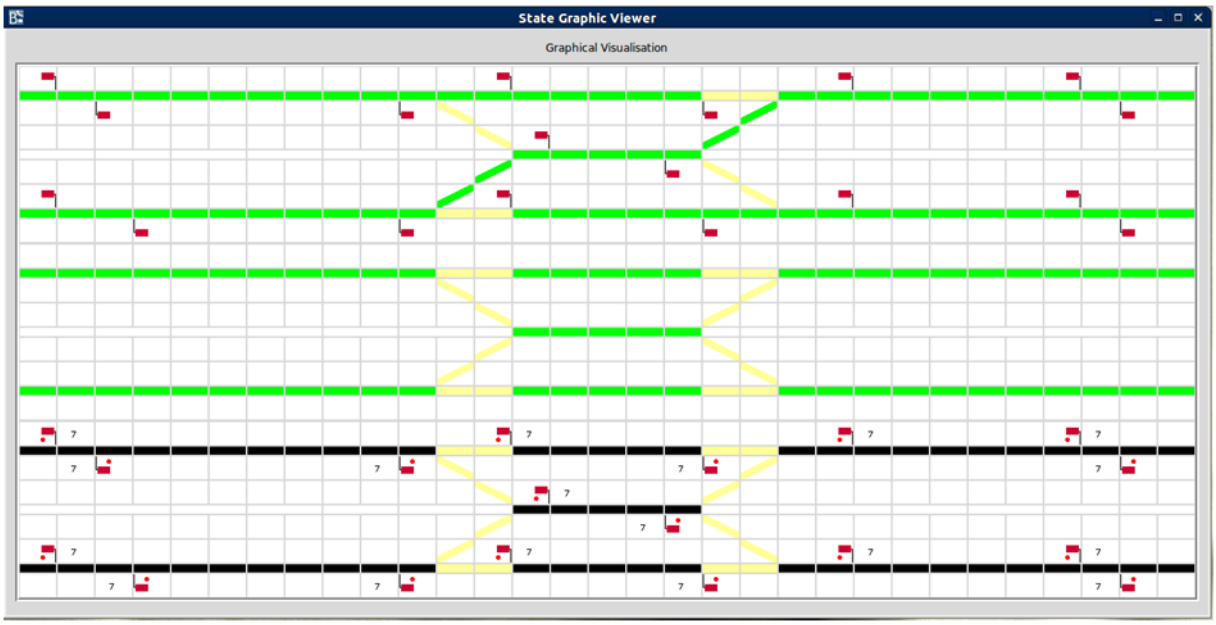
The model of the environment involves invariants formalising the safety requirements presented above (cf. section §4.2.1) and events controlling the forward and backward movements of trains (with possible sliding and abnormal shunting of track sections), the movements of points, the aspect of signals and failure and repair of sensors. The model of the interlocking system only involves events corresponding to signal and point commands. The model of the dynamic controller involves the events reacting to changes of the state of trackside sensors and to interlocking commands.

ProB allows the user to display a graphical representation of the state of the animated model. The user can therefore visualise graphically the evolution of the state of its model during an animation session. The graphical representation is a matrix of graphical icons controlled by the value of the state of the model and displayed in a separate window.

The figure below shows the graphical representation of the state of the complete model and more precisely of a particular situation (selected by the person animating the model) at the start-up of the system.

We notice three graphs representing a track involving two mainlines, one crossover made of six points allowing trains going from one mainline to the other and extremity route signals.

The graph at the top represents the actual state of the track. The graph in the middle represents the state of the sensors of the track sections and the points. The graph at the bottom represents the state of the track as seen by both the dynamic controller and the interlocking system.

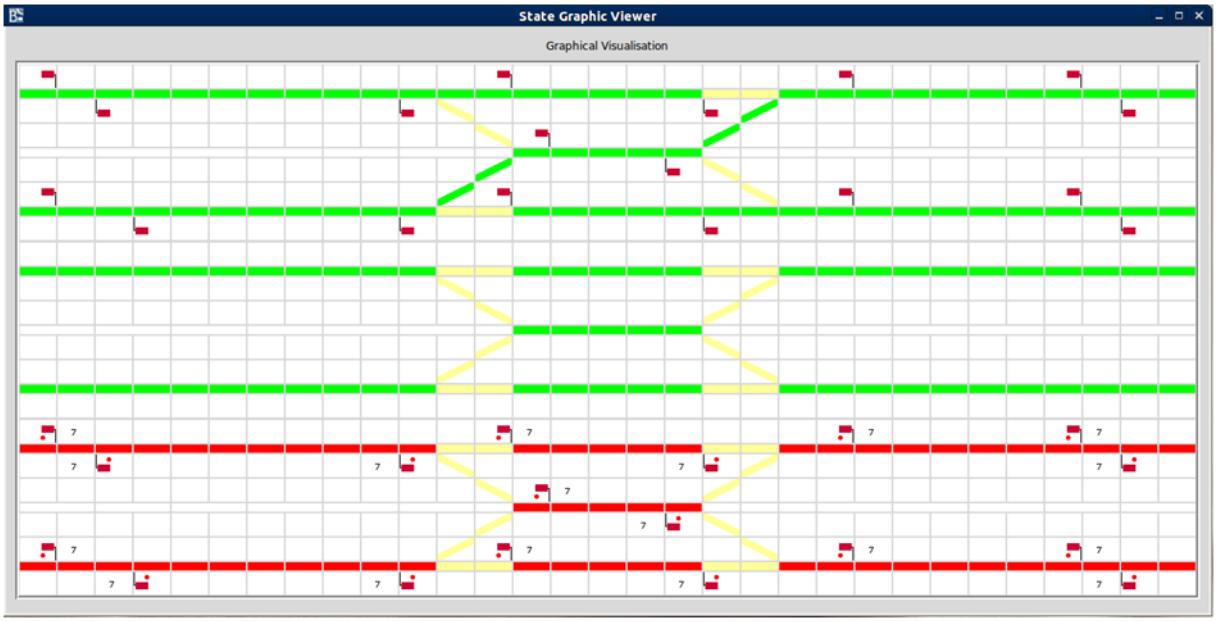


A point is represented by a single line section followed by two divergent line sections that represent the two branches of the point. A light yellow line section indicates that branch is not accessible i.e. that the point is not in that position. Thus, an uncontrolled point is represented by a single line section followed by two diverging light yellow line sections. A green line section indicates that that track section is free. A black line section indicates that that track section is occupied. A red line section (not present in the above figure) indicates that the track section is booked for a train.

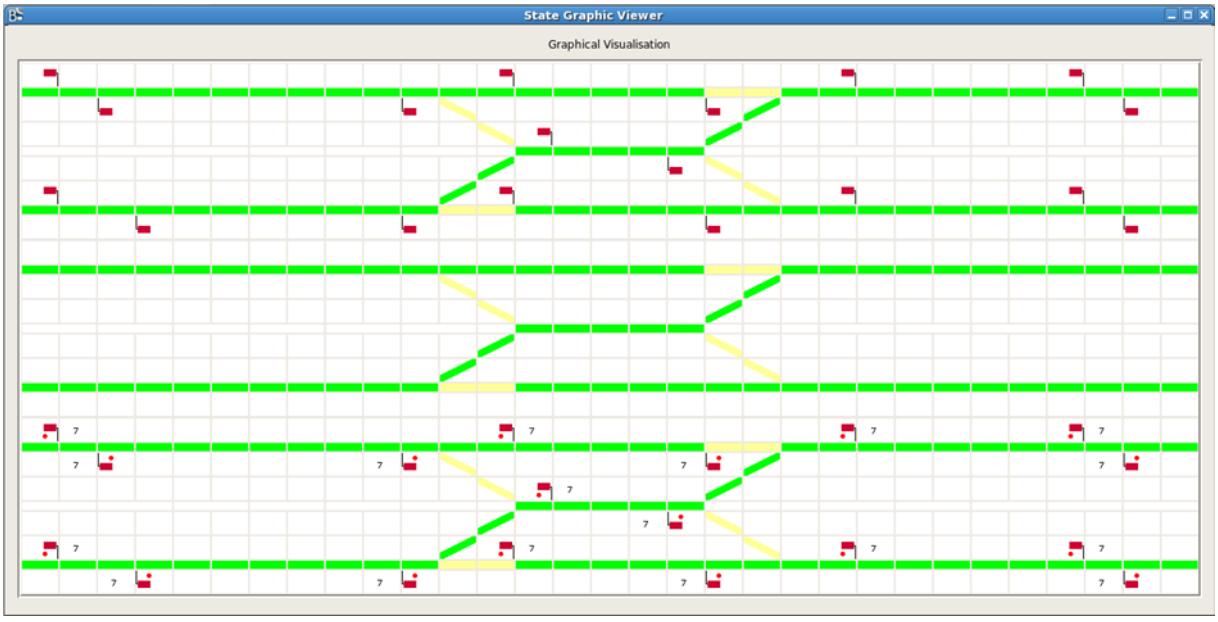
A red box above a straight line represents a closed (or restrictive) route signal for trains moving from right to left (odd direction). A red box beneath a straight line represents a closed route signal for trains moving from left to right (even direction). Signals may be also green which indicates they are open (or permissive). A red dot between the signal and the line indicates that trains will not pass the signal. If the dot is green this indicates that a train may pass the signal even if the signal is restrictive. Finally the figure in front of a signal represents the value, in seconds, of a timer counting the duration of occupancy of the approach locking area of the signal that commands the release of blocks behind the signal.

Thus, in the figure above the top graph indicates that all points are actually in left position, that no track section is actually occupied and that all signals are closed. The middle graph indicates that the track section sensors (track circuits) detect the actual state of track sections but that the point sensors do not detect the actual position of the points. The bottom graph indicates that the dynamic controller considers that all points are uncontrolled, that all track zones are occupied, that all route signals are closed and that approach locking areas of signals are occupied.

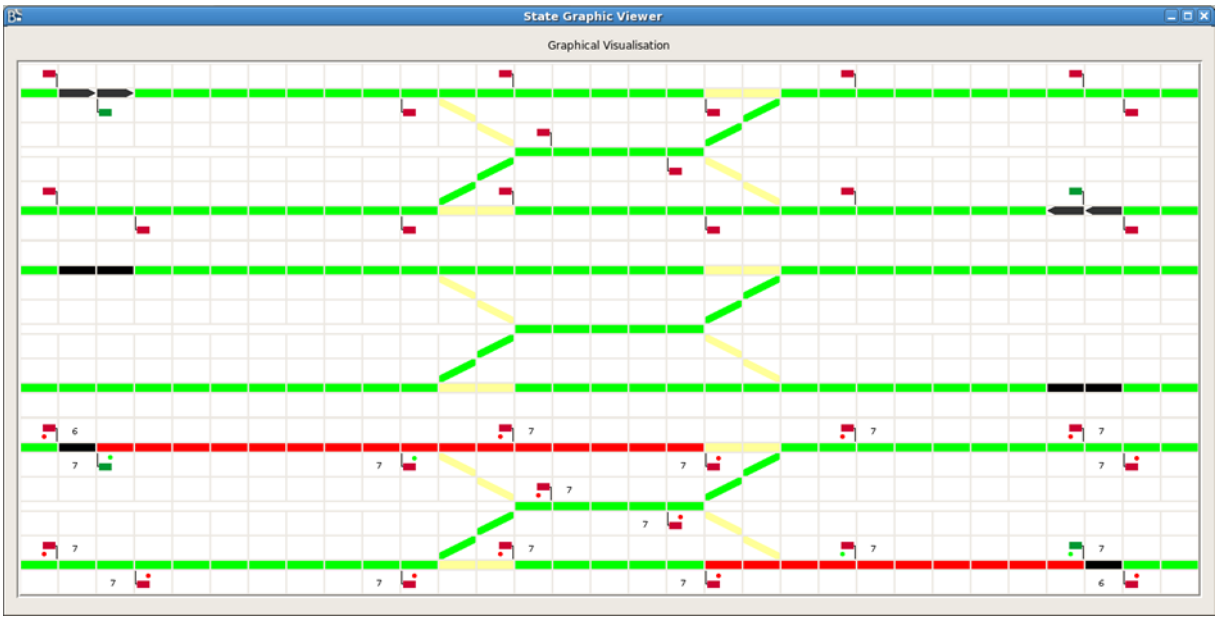
The figure below shows the state of the dynamic controller after initialisation and before it has got the state of sensors. So the dynamic controller considers all track sections booked, all points uncontrolled and all signals closed.



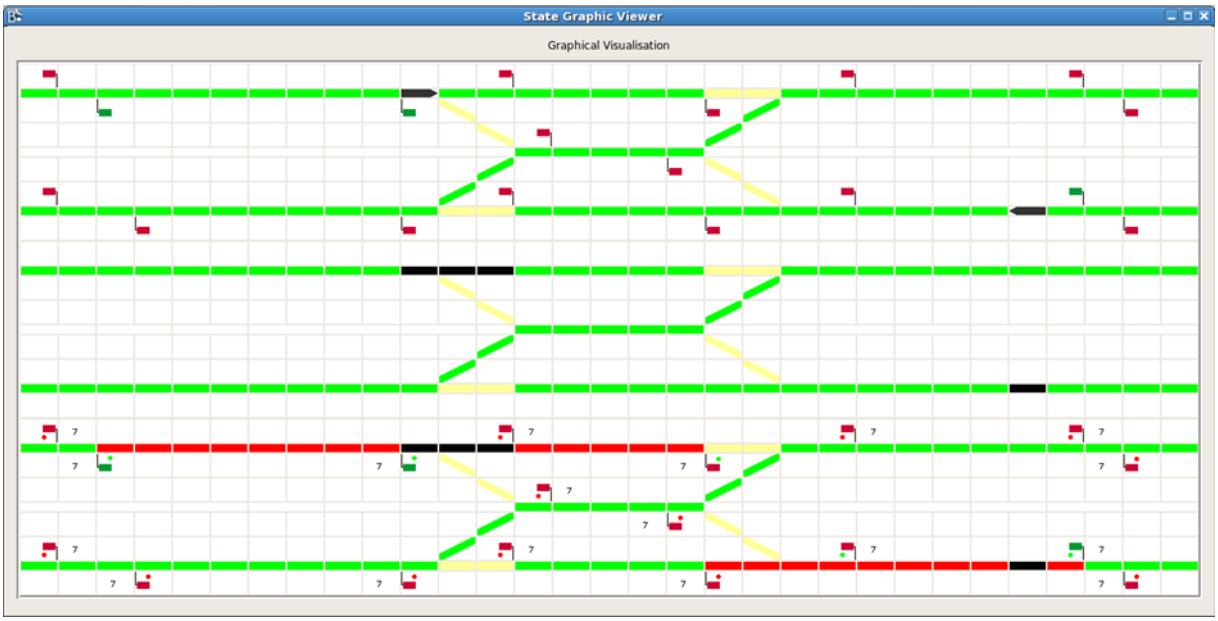
The figure below shows the state of the dynamic controller after the point sensors have being fixed and the dynamic controller has got the state of the sensors. So the dynamic controller considers all track sections free and all points controlled in left position.



The figure below shows that two routes with their overlaps have been booked (red line sections), that their entry signals are open (green boxes) and that two trains has entered these routes (black line sections). It shows also that the dynamic controller is not aware yet that the trains passed the entry signals of the routes.



Notice that the black line sections of the top graph are oriented whilst those of the two other graphs are not. This is because the environment is aware of the travel direction of the trains whilst the dynamic controller knows only the state of the sensors occupied/not occupied. A consequence of this, and the fact that there is a single track circuit for a point, is that when a train runs over a point the environment will occupy only the track sections of the point effectively covered by the train whilst the sensors and the dynamic controller will consider that all linked track sections of the point are occupied. As shown in the figure below.



A paramount benefit of the graphical animation of the model is that we can analyse the behaviour of the dynamic controller in the presence of failures of trackside equipment or of unexpected behaviour of trains. Thus we can analyse the consequences of those failures on the safety of the system. In particular, we analysed the consequences of failures of track circuits, point sensors and shunting, on booking and releasing track sections, switches and overlaps. We analysed also the consequences of a train passing a closed signal.

Through the ProB animation we found errors in the model, and this allowed us to correct it and make it more realistic, and more important, in the specification of the release of track sections and overlaps that may lead to unsafe situations.

5. Transition from Event-B to B

The purpose of this task is to define and experiment with the principles of a proved transition from an Event-B system model into sequential programs able to be translated into executable code. We also present our initial plans for translating Event-B to Classical-B since this will be important for linking the system-level safety analysis using Event-B with the existing software development process with Classical-B used by Alstom.

The input of the task is the system model of the dynamic controller. The outputs of the task are the abstract software model of the dynamic controller and a transition guide from an Event-B system model to a B software model.

5.1 ADVANCE Concepts or Tools Applied

An important aspect of the case study is the definition of a method for the proved transition from an Event-B system model into sequential programs able to be translated into executable code.

J.-R. Abrial presents in [1] an approach of development of sequential programs from Event-B models, but it is not yet supported by the Rodin platform. We shall experiment it when it will be supported.

Instead, we investigated the refinement of an Event-B system model into a B software model able to be refined itself into sequential programs. We provide feedback on this work in the following sections

5.2 Feedback on Transition from Event-B to B

As said before we developed an Event-B machine of the dynamic controller and, for animation reasons, we translated that Event-B machine into a classical B machine. An event of the Event-B machine of the form:

```
EV =
ANY X WHERE
    G
THEN
    S
END
```

is kept identical in the classical B machine translation.

Then we wanted to refine that classical B machine with a classical B implementation involving an operation of the form:

```
EV =
VAR X IN
    T
END
```

where T is a sequential program.

But this cannot be a valid refinement since it is not possible to prove that the refinement operation implies the guard G of the abstract operation in all possible cases.

To do that it would be necessary to introduce in an intermediate refinement an operation with a precondition of the form:

```
EV =
PRE
  ∃ X. G
THEN
  S'
END
```

where S' refines S.

But again, this cannot be a valid refinement because it is not possible to prove that the abstract operation implies the precondition of the refinement operation.

So we decided to modify the translation of events of Event-B into operations of classical B by translating an event of Event-B of the form presented above by a B operation of the form:

```
EV =
PRE
  ∃ X. G
THEN
   ANY X WHERE
      G
   THEN
      S
   END
END
```

And then it is possible to prove that a classical B implementation like the one presented above refines this operation. Furthermore this translation corresponds to our intuitive comprehension of the behaviour of an event and to the behaviour of animation implemented by ProB. That is to say, that the event can be fired or animated only when there exist values X satisfying the guard G.

6. Feedback on ADVANCE Tools

So far we have experienced with several concepts and tools of the ADVANCE project: requirements definition following J.-R. Abrial's recommendations, system modelling and proof with Event-B and Rodin, model animation with ProB and transition from system models (Event-B) to software models (B).

As regarding Event-B modelling and proof, this work allowed us to highlight that the theory plug-in is essential because it simplifies the model writing and modification and facilitates the proof activity.

We have the same conclusion about the animation with ProB. It disclosed two problems in our model and helped us to render it more accurate and relevant. Unfortunately, for the time being, Rodin and ProB are not at the same level of development. ProB does not deal with theories and some expressions undermine its performances. To bypass the last problem, we rewrote these expressions so they can be animated. However, from the beginning of our study we have designed our Event-B model to facilitate the proof activity and this rewriting goes against this effort. So, an important step forward will be to bring these tools at the same level of development so that it will be possible to animate with ProB all Event-B models. To achieve this goal, it may be necessary to introduce some guidelines or writing rules or have for example two theories, one for proof and one for animation. We have exactly the same concerns with axioms of the Event-B model.

As regarding the transition from system modelling to software modelling we believe that the schema to move from Event-B to B presented above introduces an interesting distinction between the phases of specification and implementation of the system. The former aims to enrich gradually an Event-B model and define the actual operational events of the system and the latter aims to reify gradually the translation to B operations of these operational events until obtaining an executable code.

Some proof of concept activities could not be completed because of tooling limitations which is outside the control of WP1. Work started of using Rodin but then switched to AtelierB because of the lack of integration of ProB with the Theory plug-in. The consequence of this is that other plug-ins of Rodin could not be used (e.g., use of ProR for the traceability of the refinement plan, model-based testing). These tooling issues are being dealt with by on-going improvements to the tools and in the full deployment phase we shall use the latest Rodin and assess the suitability of the new tool development from WP3 and WP4

We shall port to Rodin the B model animated with ProB. This will give us the opportunity to experiment with the component composition plug-in of Rodin and the theory plug-ins of Rodin that provide data structures and operators available in the B language, and used in our model, that do not exist in Event-B (sequence data structure and its related operators and closure relation operator). Support for these operators in Rodin is currently being developed through he Theory plug-in and through integration of the Theory plug-in with ProB.

7. Future Work in WP1

The proof of concept phase of WP1 has demonstrated the feasibility of using Event-B for modelling and verification of system-level properties of railway interlocking. The next stages of WP1 will involve the development of a strategy for certification using ADVANCE methods and tools together with a more complete treatment of the interlocking through refinement and decomposition down to the architectural level. Here we make some initial observations about the safety analysis and the refinement plan.

7.1 Safety Analysis

The purpose of this task is twofold. First, using fault tree analysis or failure mode effects analysis, ensure that all safety requirements have been identified and, if this is not the case, introduce the missing ones. Second, arrange safety requirements according to the refinement order of Event-B models in order to ensure that the system level requirements have been correctly and completely decomposed and allocated to lower level subsystems.

The inputs of this task are the preliminary hazard analysis, the requirements document and the refinement plan. The outputs of this task are the set of safety requirements and the coverage by the formal model of those requirements.

The safety analysis will be dealt with by Task 1.3 with the support of an Alstom RAMS expert and University of Southampton. As well as using standard safety analysis analysis methods, we will explore the use of an approach based on STPA and Event-B being developed in WP5.

7.2 Refinement Plan

The purpose of the refinement plan task is to identify the levels of refinement necessary to move from an abstract but incomplete model of the dynamic controller to an abstract but complete model.

The input of this task is the requirements document. The output of this task is the refinement plan document which presents and justifies each level of refinement.

The levels of refinement will be created according to the progressive introduction of concepts and the progressive modelling of requirements. However, attention should be paid to create the minimum number of levels of refinement in order to facilitate the maintenance of the model.

Allocation of requirements to refinement levels will be managed with the ProR tool developed by the University of Düsseldorf [2].

8. References

[1] Modeling in Event-B – System and Software Engineering. J.-R. Abrial. © Cambridge University Press 2010.
[2] ProR – Requirement Engineering Platform. http://www.eclipse.org/rmf/pror/
[3] ProB – The ProB Animator and Model Checker. http://www.stups.uni-duesseldorf.de/ProB © Formal Mind
[4] AtelierB – http://www.atelierb.eu/en/ © Clearsy