



DELIVERABLE D1.3 – WORK PACKAGE 1

INTERMEDIATE REPORT ON APPLICATION ON RAILWAY DOMAIN

ADVANCE

Grant Agreement:	287563
Date:	October 21 st 2013
Author:	Fernando Mejia, Alstom; Minh-Thang Khuu, Systerel
Status:	
Reviewer:	M. Leuschel; M. Butler
Reference:	D1.3
Issue:	1
Partners / Clients:	
 SEVENTH FRAMEWORK PROGRAMME FP7 Framework Programme	
 European Union	

Consortium Members:

 UNIVERSITY OF Southampton	 Critical software	 ALSTOM	 Systerel Safe real-time solutions	 Heinrich Heine UNIVERSITÄT DÜSSELDORF
University of Southampton	Critical Software Technologies	Alstom Transport	Systerel	Heinrich Heine Universität Düsseldorf

Contents

1.	Introduction	4
2.	Proof of the IXL-DC Event-B model	4
3.	Testing of the IXL-DC Event-B model.....	16
4.	Intermediate assessment of ADVANCE tools.....	18
4.1	Event-B.....	19
4.2	Rodin.....	19
4.2.1	Theory plug-in	20
4.3	ProB	20
4.4	Co-simulation	21
5.	Contribution of ADVANCE Methods and Tools to Alstom’s certification process	21
5.1	Certification framework.....	21
5.1.1	CENELEC standard EN 50126	21
5.1.2	CENELEC standard EN 50129	23
5.2	Introduction of ADVANCE Methods and Tools in Alstom’s system development process ...	28
5.2.1	System Definition	28
5.2.2	Preliminary Hazard Analysis	28
5.2.3	Requirements Specification.....	29
5.2.4	Requirements Verification.....	30
5.2.5	System Hazard Analysis	31
5.2.6	Architecture Specification	31
5.2.7	Architecture verification.....	32
5.2.8	Interface Hazard analysis.....	32
5.2.9	Sub-systems development	32
5.2.10	Sub-system safety cases consolidation	33
5.2.11	System integration	34
5.2.12	Safety integration verification	34
5.2.13	System validation	34
5.2.14	System safety case consolidation	34
5.2.15	System acceptance.....	34
6.	Concluding.....	35
	References.....	36

List of figures

Figure 3.1. Envisioned IXL-DC code generation process and test environment architecture.....	16
Figure 3.2. Implemented IXL-DC code generation process and test environment architecture.....	18
Table 5.1. Safety Tasks for each phase defined in EN50126	23
Figure 5.2. Example of design and validation life-cycle defined in EN50129	25
Figure 5.3. Alstom’s system development process compliant with EN50129	28
Figure 5.4. Requirements Specification formal model development process	30
Figure 5.5. Architecture Specification formal model development process.....	32
Figure 5.6. Alstom’s Software development process with Classical-B	33

1. Introduction

The first period of the project (October 2101-September 2012) was devoted to a feasibility study of the formal development in Event-B of an interlocking dynamic controller (IXL-DC). That is to say the ability to construct with ADVANCE methods and tools an Event-B model of an independent component that would control commands of the interlocking system (IXL) in order to ensure protection of trains against face-to-face collisions, flank collisions and derailment. Within this period we defined the requirements of the IXL-DC, constructed an abstract model of a signalling system including a model of the IXL-DC and studied the transition from an Event-B model to a Classic-B model for the development of its software. For this study we used methods and tools provided by ADVANCE: the Rodin platform and its Theory plug-in to construct the system model and demonstrate some of its proof obligations and ProB to ensure by interactive animation that the behaviour of the model is indeed the one we imagined. We presented the activities of the first period and their results in deliverable D1.2.

This deliverable presents the activities undertaken during the second period of the project (October 2012-September 2013). This period was devoted to demonstrate the validity of the IXL-DC model constructed within the previous period and to define a system development process involving development of formal models and compliant with applicable certification standards in railway domain.

Sections 2 and 3 of the deliverable describe the validation of the IXL-DC model. Section 2 describes the steps followed to prove the mathematical correctness of the model. Their purpose is to control the number and the complexity of proof obligations by means of stepwise enrichment of a fairly abstract initial model. This method is the one promoted by ADVANCE and has already demonstrated its effectiveness. Section 3 describes the process followed to test the behavioural correctness of the IXL-DC model. The purpose is to verify under realistic operating conditions that IXL-DC is neither too permissive nor too restrictive. The process includes the generation of a compiled executable version of the IXL-DC model and the reuse of factory acceptance tests of an actual signalling system.

In section 4 of the deliverable we give an intermediate assessment of the methods and tools used so far for our case study. For each of them we analyse its ability to be adopted by industry according to technical, maintenance and sustainability criteria.

The fifth and final section of the deliverable presents the system development process including development of formal models with ADVANCE methods and tools that Alstom might adopt. The first part of the section presents the regulatory framework, that is to say, the requirements on the system development process made by the European CENELEC certification standards and applicable to the railway industry. The second part of the section presents the steps of the process and explains how the activities related to the development of formal models with ADVANCE methods and tools contribute to meet the requirements of certification standards.

2. Proof of the IXL-DC Event-B model

The proof of an Event-B model is considerably easier if the model is constructed by successive refinements that enrich progressively an abstract initial model. The first step of this process consists of creating a refinement plan that defines and explains the number, the nature and the purpose of the refinements that will be done.

This section presents the refinement plan for the Event-B model of the IXL-DC. The objective of this plan is to define the refinement steps in such a way that the last refinement is equivalent to the IXL-DC model created in the first period.

Event-B model

The model must incorporate certain IXL principles, the environment and IXL-DC. The abstract level describes a high-level view of the whole system. As the refinement goes further, closer views of the

system are given. The IXL-DC plays the role of a filter to IXL commands, including commands of signal aspects, of point movement, of directions on blocks and commands of overlap locked signals.

The model must take into account the changes of the environment status. These changes are modelled by events including those of train movement (train advancing, train entering or leaving the considered track network, train turnaround), of train shunting on track circuit, of point movement and of signal aspect changes.

Modelling context: the model is based on a general configuration of track network and related devices. These data consist of disposition of blocks, points and signals, the definition of timers and secondary detection devices (SDD). The correctness of these data is out of the scope of our modelling activity. Formalised assumptions on these data are used as entry elements of the model.

The initial model:

Modelling rationale: The previous deliverable D1.2 has presented an initial model of the IXL-DC system and the usage of the Theory plug-in. In this first model, trains on the track network are modelled by chains of oriented blocks and the non-collision properties are expressed by the compatibility of these areas:

- (0_1) Points under a train must be well positioned.
- (0_2) Every two train block chains are compatible.

The above two properties can be expressed within a block chaining structure and operators chain_on_graph and is_ch_compatible) defined by using the Theory plug-in. For example, let v_next_blocks be the dynamic track network,

- v_trains : t_train +-> chain_on_graph(v_next_blocks)
- is_ch_compatible(ran(v_train), c_incompatible_blocks)

Modelling context data: This initial model uses the following data of track network configuration:

- Static and dynamic chaining of blocks;
- Configuration of fouling blocks. Note that two oriented blocks are incompatible if they are either fouling blocks or opposite blocks. A block is incompatible with itself.
- Turn around blocks.

Events:

Events	Description	Guards and actions
train_front_move	Train front advances one block	<u>Guard</u> : the block added to the train front must be compatible with other train chains. <u>Action</u> : update train chaining.
train_rear_move	Train rear advances one block	<u>Action</u> : update train chaining.
train_turnaround	Train turns around	<u>Guard</u> : train must be in the turnaround block zone; and blocks in the turnaround block chain must be compatible with other train chains. <u>Action</u> : update train chaining.
train_leave_area	Train leaves the considered network area.	<u>Action</u> : update train on the network area.
train_enter_into_area	Train enters into the network area.	<u>Guard</u> : extremity blocks where trains enter must be compatible with other train chains. <u>Action</u> : update train chaining.
signal_permissive_aspect		<u>Guard</u> : the signal must be in restrictive aspect. <u>Action</u> : update signal aspect.

signal_restrictive_aspect		<u>Guard</u> : the signal must be in permissive aspect. <u>Action</u> : update signal aspect.
point_move_to_unknown	Point moves from left or right position to unknown position.	<u>Guard</u> : The point must not be under any train. <u>Action</u> : update point position.
point_move_to_right	Point moves from unknown position to right position.	<u>Action</u> : update point position.
point_move_to_left	Point moves from unknown position to left position.	<u>Action</u> : update point position.

First refinement:

Refinement rationale: In this refinement, protected areas are introduced. The notion of protected area is more general than that of train movement authorization zone: a protected area is not necessarily assigned to a train. Protected areas must ensure non-collision and non-derailment of trains. Safety properties can be expressed as follows:

- (1_1) A train is covered by a protected area
- (1_2) Points in a protected area must be well positioned
- (1_3) Protected areas must be pairwise compatible

The property (1_1) is a link property. As trains are covered by protected area, properties (0_1) and (0_2) in the abstract machine can be refined into the properties (1_2) and (1_3). In other words, the abstract properties hold whenever the concrete properties hold.

Protected areas can be implemented by chains of oriented blocks on the dynamic track network.

Modelling context data: this refinement uses the same modelling context data as that of its abstract level.

Events: Events and their abstract events are listed in the following table. Bold events are those that are first introduced in this refinement; underlined events are those that have guard strengthened or actions added/modified:

Events	Description	Guards and actions
<i>train_front_move</i> <u>abstract</u> : train_front_move	Cf. abstract	<u>Guard</u> : the block added to the train front must be in the train protected area.
train_rear_move	Cf. abstract	
<i>train_turnaround</i> <u>abstract</u> : train_turnaround	Cf. abstract	<u>Guard</u> : protected area of train must be in the turnaround block zone.
train_leave_area	Cf. abstract	
<i>train_enter_into_area</i> <u>abstract</u> : train_enter_into_area	Cf. abstract	<u>Action</u> : create a protected area covering the train.
signal_permissive_aspect	Cf. abstract	
signal_restrictive_aspect	Cf. abstract	
<i>point_move_to_unknown</i> <u>abstract</u> : point_move_to_unknown	Cf. abstract	<u>Guard</u> : The point must not be in a protected area.

point_move_to_right	Cf. abstract	
point_move_to_left	Cf. abstract	
reduce_rear_protected_area	Reduce rear of a protected area	<u>Guard</u> : rear of the protected area must not be in a train chaining. <u>Actions</u> : update protected areas.
reduce_front_protected_area	Reduce front of a protected area	<u>Guard</u> : front of the protected area must not be in a train chaining. <u>Actions</u> : update protected areas.
remove_protected_area	Remove a protected area	<u>Guard</u> : protected area reduced to a block and this block must not be in a train chaining. <u>Actions</u> : update protected areas.
extend_protected_area	Extend a protected area by a set of blocks.	<u>Guard</u> : the extended blocks must be compatible with other protected areas. <u>Actions</u> : update protected areas.
split_protected_area	Split a protected area at a given block into two protected areas not containing the splitting block.	<u>Guard</u> : The given block must not be in a train chaining and must be neither the rear nor the front of the protected area. <u>Actions</u> : update protected areas.

Second refinement:

Refinement rationale: This refinement introduces the filtered commands of points into the model and refines the movement of points. The movement of a point from left (right) position to unknown position implies that its conjugated points are also moved to unknown position. A point is in moving from unknown to left (right) position implies that its conjugated points are in moving from unknown position to their corresponding positions. A point is moved from unknown position to left (right) if it was moving to left (right). A point moves only when commanded. Safety properties are refined as follows:

- (2_0) Point commands must not be contradictory (point is commanded in only one position). Furthermore conjugated points of commanded points must not be contradictory (a point is associated to only one position to move to).
- (2_1) A commanded point and its conjugated point must NOT belong to a protected area.

When the properties 2_0 and 2_1 are verified, properties 1_2 and 1_3 are also verified.

Modelling context data: In addition to context data used in the abstract level, this refinement uses the following context data:

- Conjugated point of a point

Events: Events and their abstract events are listed in the following table. Bold events are those that are first introduced in this refinement; underlined events are those that have guard strengthened or actions added:

Events	Description	Guards and actions
train_front_move	cf. abstract	
train_rear_move	cf. abstract	
train_turnaround	cf. abstract	
train_leave_area	cf. abstract	
train_enter_into_area	cf. abstract	<u>Guard</u> : In IXL data configuration, there is

		no point on extremity blocks. Thus extremity blocks contain no point in point command set.
signal_permissive_aspect	cf. abstract	
signal_restrictive_aspect	cf. abstract	
reduce_rear_protected_area	cf. abstract	
reduce_front_protected_area	cf. abstract	
remove_protected_area	cf. abstract	
split_protected_area	cf. abstract	
extend_protected_area (<u>abstract</u> : extend_protected_area)	cf. abstract	<u>Guard</u> : extended blocks must not contain any point which is in the point commands or in the conjugated point set of a commanded point.
point_move_to_unknown (<u>abstract</u> : point_move_to_unknown)	cf. abstract	<u>Guard</u> : point must not belong to any protected area; and there is a point command to the opposite position.
point_move_to_right (<u>abstract</u> : point_move_to_right)	cf. abstract	<u>Guard</u> : point is moving to right and there is a point command to right position. <u>Actions</u> : update point position (right).
point_move_to_left (<u>abstract</u> : point_move_to_left)	cf. abstract	<u>Guard</u> : point was moving to left and there is a point command to left position. <u>Actions</u> : update point position (left).
point_is_moving_to_right	Point is moving from unknown to right position.	<u>Guard</u> : point is unknown and there is a point command to right position. <u>Actions</u> : update point is moving.
point_is_moving_to_left	Point is moving from unknown to left position.	<u>Guard</u> : point is unknown and there is a point command to left position. <u>Actions</u> : update point is moving.
cmd_point_to_right	Send a point command.	<u>Guard</u> : point and its conjugated points must not belong to any protected area. <u>Actions</u> : add point command.
cmd_point_to_left	Send a point command.	<u>Guard</u> : point and its conjugated points must not belong to any protected area. <u>Actions</u> : add point command.
no_cmd_point	Remove a point command.	<u>Actions</u> : remove a point command.

Third refinement:

Refinement rationale: This refinement introduces traffic direction commands and all booked blocks. All booked blocks include booked blocks and overlap booked blocks. A traffic direction can be set on a block, only if the block was already booked and had no direction. In our model, we examine booked blocks from the moment directions have first been set on them. Therefore points in point commands or in their conjugated points have no underlying booked blocks. The refinement is based on the following safety properties:

- (3_1) Traffic direction is defined only on a booked block.
- (3_2) Protected areas are covered by booked blocks.
- (3_3) On all booked blocks, no point is commanded nor its conjugated point.
- (3_4) All booked blocks are compatible.

The properties (2_0) and (2_1) hold when (3_2), (3_3) and (3_4) hold.

Modelling context data: this refinement uses the same modelling context data as that of its abstract level.

Events: Events and their abstract events are listed in the following table. Bold events are those that are first introduced in this refinement; underlined events are those that have guard strengthened or actions added:

Events	Description	Guards and actions
train_front_move	cf. abstract	
train_rear_move	cf. abstract	
<i>train_turnaround</i> (<u>abstract</u> : train_turnaround)	cf. abstract	<u>Guard</u> : there are traffic direction commands on turnaround blocks.
train_leave_area	cf. abstract	
train_enter_into_area	cf. abstract	<u>Guard</u> : the extremity block must be compatible with other booked blocks. <u>Action</u> : add the extremity block to booked blocks and set the traffic direction on it.
signal_permissive_aspect	cf. abstract	
signal_restrictive_aspect	cf. abstract	
reduce_rear_protected_area	cf. abstract	<u>Action</u> : release booked block.
reduce_front_protected_area	cf. abstract	<u>Action</u> : release booked block.
remove_protected_area	cf. abstract	<u>Action</u> : release booked block.
split_protected_area	cf. abstract	<u>Action</u> : release booked block.
extend_protected_area	cf. abstract	
<i>point_move_to_unknown</i> (<u>abstract</u> : point_move_to_unknown)	cf. abstract	<u>Guard</u> : point and its conjugated point must not belong to any booked blocks.
point_move_to_right	cf. abstract	
point_move_to_left	cf. abstract	
point_is_moving_to_right	cf. abstract	
point_is_moving_to_left	cf. abstract	
<i>cmd_point_to_right</i> (<u>abstract</u> : cmd_point_to_right)	cf. abstract	<u>Guard</u> : point and its conjugated point must not belong to any booked blocks.
<i>cmd_point_to_left</i> (<u>abstract</u> :cmd_point_to_left)	cf. abstract	<u>Guard</u> : point and its conjugated point must not belong to any booked blocks.
no_cmd_point	cf. abstract	
set_traffic_direction	Set a traffic direction on a block.	<u>Guard</u> : There is a traffic direction command on the block and, the block must either have been booked or be compatible with all other booked blocks. <u>Action</u> : update the booked blocks and traffic directions on blocks.
release_traffic_direction	Release the traffic direction of a block.	<u>Action</u> : release the traffic direction of the block.
cmd_traffic_direction	Set a traffic direction command.	<u>Action</u> : update the traffic direction commands.
set_traffic_direction_reversal	Set traffic direction on blocks.	<u>Guard</u> : there is a traffic direction command on the block and, the block must have no traffic direction and their opposite blocks must have been booked.

		<u>Action</u> : update the traffic direction on blocks and booked blocks.
no_cmd_traffic_direction	Unset a traffic direction command.	<u>Action</u> : update the traffic direction commands.
cmd_overlap_booked_block	Book overlap blocks.	<u>Guard</u> : the blocks must be compatible with all other booked blocks. <u>Action</u> : update the overlap booked blocks.

Fourth refinement:

Refinement rationale: this refinement introduces track circuits in order to strengthen guards of events. The controller does not know anymore the position of trains, but has a fuzzier picture based on track circuits that are shunted by trains.

Modelling context data: In addition to context data used in the abstract level, this refinement uses the following context data:

- Underlying SDD of blocks.

Events: Events and their abstract events are listed in the following table. Underlined events are those that have guard strengthened or actions added:

Events	Description	Guards and actions
<u>train_front_move</u> (<u>abstract</u> : train_front_move)	cf. abstract	<u>Action</u> : update train shunt status.
<u>train_rear_move</u> (<u>abstract</u> : train_rear_move)	cf. abstract	<u>Action</u> : update train shunt status.
<u>train_turnaround</u> (<u>abstract</u> : train_turnaround)	cf. abstract	<u>Action</u> : update train shunt status.
<u>train_leave_area</u> (<u>abstract</u> : train_leave_area)	cf. abstract	<u>Action</u> : update train shunt status.
<u>train_enter_into_area</u> (<u>abstract</u> : train_enter_into_area)	cf. abstract	<u>Action</u> : update train shunt status.
signal_permissive_aspect	cf. abstract	
signal_restrictive_aspect	cf. abstract	
reduce_rear_protected_area	cf. abstract	<u>Guard</u> : Underlying SDD of the block must be clear.
reduce_front_protected_area	cf. abstract	<u>Guard</u> : Underlying SDD of the block must be clear.
remove_protected_area	cf. abstract	<u>Guard</u> : Underlying SDD of the block must be clear.
split_protected_area	cf. abstract	<u>Guard</u> : Underlying SDD of the block must be clear.
extend_protected_area	cf. abstract	
point_move_to_unknown	cf. abstract	
point_move_to_right	cf. abstract	
point_move_to_left	cf. abstract	
point_is_moving_to_right	cf. abstract	
point_is_moving_to_left	cf. abstract	
cmd_point_to_right	cf. abstract	

cmd_point_to_left	cf. abstract	
no_cmd_point	cf. abstract	
set_traffic_direction (<u>abstract</u> : set_traffic_direction)	cf. abstract	<u>Guard</u> : underlying SDD of blocks must be clear.
release_traffic_direction	cf. abstract	
cmd_traffic_direction	cf. abstract	
set_traffic_direction_reversal (<u>abstract</u> : set_traffic_direction_reversal)	cf. abstract	<u>Guard</u> : underlying SDD of blocks must be occupied.
no_cmd_traffic_direction	cf. abstract	
cmd_overlap_booked_block	cf. abstract	

Fifth refinement:

Refinement rationale: This refinement introduces signals and related timers. In this machine, protected area formation is defined: when a signal turns to permissive aspect, a protected area is extended from the downstream block of the signal until the upstream block of the next signal or possibly sliding blocks of the next signal. Protected blocks are removed when blocks are un-booked.

Modelling context data: In addition to context data used in the abstract level, this refinement uses the following context data:

- Upstream and downstream of signals.
- Approach section of signals.
- Turnaround blocks of signals.
- Sliding sections of signals.
- Timers associated to signals.

Events: Events and their abstract events are listed in the following table. Bold events are those that are first introduced in this refinement; underlined events are those that have guard strengthened or actions added:

Events	Description	Guards and actions
train_front_move	cf. abstract	
train_rear_move	cf. abstract	
train_turnaround	cf. abstract	
train_leave_area	cf. abstract	
train_enter_into_area	cf. abstract	
<u>signal_permissive_aspect</u> (<u>abstract</u> : signal_permissive_aspect)	cf. abstract	<u>Guard</u> : there is a command of signal to permissive aspect.
<u>signal_restrictive_aspect</u> (<u>abstract</u> : signal_restrictive_aspect)	cf. abstract	<u>Guard</u> : there is a command of signal to restrictive aspect.
<u>release_signal_instantaneously</u> (<u>abstract</u> : reduce_rear_protected_area, remove_protected_area)	cf. abstract	<u>Guard</u> : the block is the downstream block of the signal. Underlying SDD of approach section of the signal must be clear.
<u>release_block</u>	cf. abstract	<u>Guard</u> : the block is NOT the downstream

(<u>abstract</u> : reduce_rear_protected_area, remove_protected_area)		block of the signal.
release_overlap_block (<u>abstract</u> : reduce_rear_protected_area, remove_protected_area)	cf. abstract	<u>Guard</u> : the block belongs to the sliding section of the signal and its previous block is not booked. <u>Action</u> : update overlap locked command on the signal.
release_signal_by_timer_on_overlap (<u>abstract</u> : reduce_front_protected_area, split_protected_area)	cf. abstract	<u>Guard</u> : the block is the downstream block of the signal. The underlying SDD of upstream block of the signal is occupied. The signal timer has expired. <u>Action</u> : update overlap locked command on the signal.
release_signal_by_sequence_TORA	First step of the sequence of signal release.	<u>Guard</u> : the block is a downstream block of signal. The underlying SDD of the block is clear, and the SDD of the previous block is occupied. The signal is permissive. <u>Action</u> : Signal sequence is set to seq_1.
release_signal_by_sequence_TORB	Second step of the sequence of signal release.	<u>Guard</u> : the block is a downstream block of signal. The underlying SDDs of the block and of the previous block are occupied, and the underlying SDD of the next block is clear. Signal sequence is seq_1. The signal is restrictive. <u>Action</u> : Signal sequence is set to seq_2.
release_signal_by_sequence_TORC (<u>abstract</u> : reduce_rear_protected_area)	Last step of the sequence of signal release.	<u>Guard</u> : the block is a downstream block of signal. The underlying SDD of the block is clear, and the SDD of the next block is occupied. Signal sequence is seq_2. The signal is restrictive. <u>Action</u> : Signal sequence is set to seq_0.
cmd_signal_permissive_aspect (<u>abstract</u> : extend_protected_area)		<u>Guard</u> : the given blocks must be between two signals and there is no other signal between these two signals. The given blocks can eventually include sliding blocks of the second signal. All these blocks must have been booked. If there are turnaround blocks of the signal in the given blocks, all trains in the turnaround area must be stopped in the exit direction. <u>Action</u> : add signal permissive command and extend protected area.
cmd_signal_permissive_aspect_toward_discrimination_area (<u>abstract</u> : extend_protected_area)		<u>Guard</u> : the given blocks must be between the signal and the discrimination area; and there is no signal between the signal and the discrimination area. All these blocks must have been booked. Underlying SDD of discrimination area is clear. <u>Action</u> : add signal permissive command and extend protected area.
cmd_signal_restrictif_aspect	Command of restrictive aspect on a signal	<u>Action</u> : add signal restrictive command.
point_move_to_unknown	cf. abstract	
point_move_to_right	cf. abstract	
point_move_to_left	cf. abstract	

point_is_moving_to_right	cf. abstract	
point_is_moving_to_left	cf. abstract	
cmd_point_to_right	cf. abstract	
cmd_point_to_left	cf. abstract	
no_cmd_point	cf. abstract	
set_traffic_direction	cf. abstract	
release_traffic_direction	cf. abstract	
cmd_traffic_direction	cf. abstract	
set_traffic_direction_reversal	cf. abstract	
no_cmd_traffic_direction	cf. abstract	
<i>cmd_overlap_locked</i> (<u>abstract:</u> cmd_overlap_booked_block)	cf. abstract	<u>Guard:</u> the given blocks must be the sliding blocks of the signal.

Sixth refinement:

Refinement rationale: This refinement introduces IXL commands. Therefore, guards of previous filtered commands are strengthened by the presence of IXL commands.

Modelling context data: this refinement uses the same modelling context data as that of its abstract level.

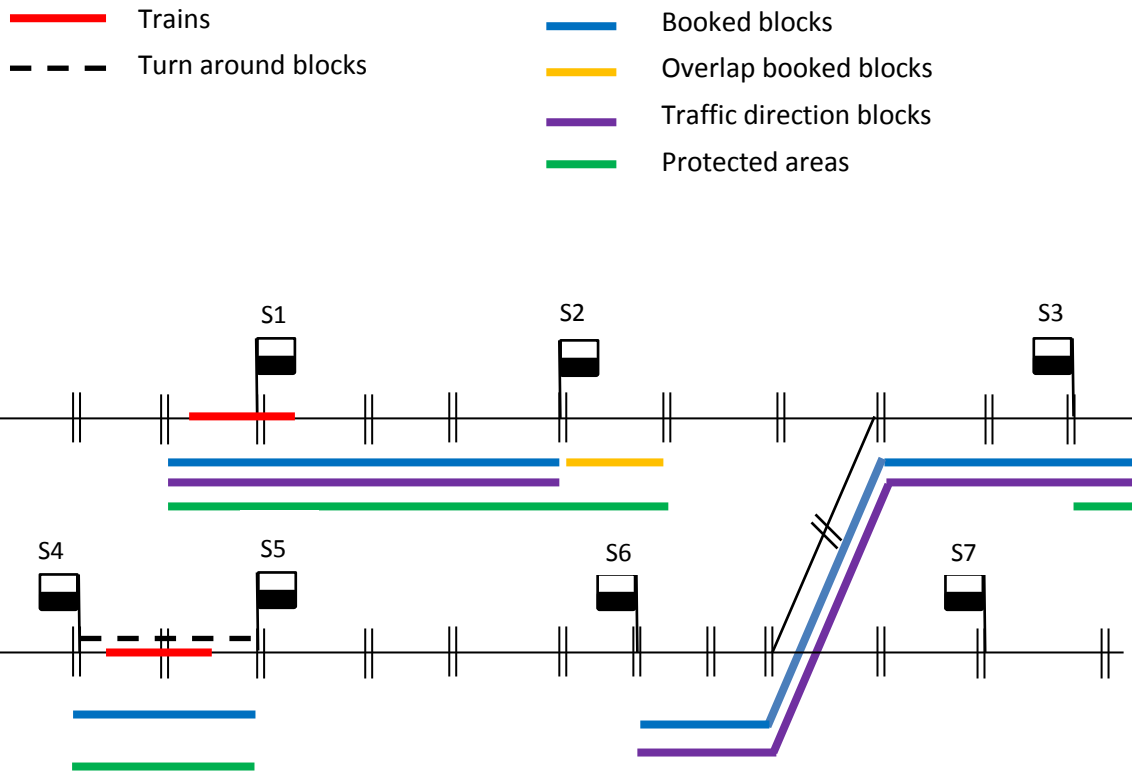
Events: Events and their abstract events are listed in the following table. Bold events are those that are first introduced in this refinement; underlined events are those that have guard strengthened or actions added:

Events	Description	Guards and actions
train_front_move	cf. abstract	
train_rear_move	cf. abstract	
train_turnaround	cf. abstract	
train_leave_area	cf. abstract	
train_enter_into_area	cf. abstract	
signal_permissive_aspect	cf. abstract	
signal_restrictive_aspect	cf. abstract	
release_signal_instantaneously	cf. abstract	
release_block	cf. abstract	
<i>release_overlap_block</i> (<u>abstract:</u> release_overlap_block)	cf. abstract	<u>Guard:</u> there is no IXL overlap locked command on the signal.
release_signal_by_timer_on_overlap	cf. abstract	
release_signal_by_sequence_TO_RA	cf. abstract	
release_signal_by_sequence_TO_RB	cf. abstract	
release_signal_by_sequence_TO_RC	cf. abstract	
<i>cmd_signal_permissive_aspect</i> (<u>abstract:</u> cmd_signal_permissive_aspect)	cf. abstract	<u>Guard:</u> there is an IXL signal permissive command.
<i>cmd_signal_permissive_aspect_t</i>	cf. abstract	<u>Guard:</u> there is an IXL signal permissive

<i>oward_discrimination_area</i> (abstract: cmd_signal_permissive_aspect_ toward_discrimination_area)		command.
<i>cmd_signal_restrictif_aspect</i> (abstract: cmd_signal_restrictif_aspect)	cf. abstract	<u>Guard</u> : there is an IXL signal restrictive command.
point_move_to_unknown	cf. abstract	
point_move_to_right	cf. abstract	
point_move_to_left	cf. abstract	
point_is_moving_to_right	cf. abstract	
point_is_moving_to_left	cf. abstract	
<i>cmd_point_to_right</i> (abstract: cmd_point_to_right)	cf. abstract	<u>Guard</u> : there is an IXL point command to right.
<i>cmd_point_to_left</i> (abstract: cmd_point_to_left)	cf. abstract	<u>Guard</u> : there is an IXL point command to left.
<i>no_cmd_point</i> (abstract: no_cmd_point)	cf. abstract	<u>Guard</u> : there is no IXL point command on the considered point.
set_traffic_direction	cf. abstract	
release_traffic_direction	cf. abstract	
<i>cmd_traffic_direction</i> (abstract: cmd_traffic_direction)	cf. abstract	<u>Guard</u> : there is an IXL traffic direction command.
set_traffic_direction_reversal	cf. abstract	
<i>no_cmd_traffic_direction</i> (abstract: no_cmd_traffic_direction)	cf. abstract	<u>Guard</u> : there is no IXL traffic direction command on the considered block.
<i>cmd_overlap_locked</i> (abstract: cmd_overlap_locked)	cf. abstract	<u>Guard</u> : there is an IXL command of overlap locked on a signal.
IXL_command_signal	An IXL signal command (permissive or restrictive)	<u>Guard</u> : the signal is not commanded on this aspect. <u>Action</u> : Update IXL signal commands.
IXL_command_point	An IXL point command (left or right)	<u>Guard</u> : there is no point command on the considered point. <u>Action</u> : Update IXL point commands.
IXL_no_command_point	Remove an IXL point command (left or right)	<u>Guard</u> : there is a point command on the considered point. <u>Action</u> : Update IXL point commands.
IXL_command_traffic direction	An IXL traffic direction command	<u>Guard</u> : there is no traffic direction command on the considered block. <u>Action</u> : Update IXL traffic commands.
IXL_no_command_traffic direction	Remove an traffic direction command	<u>Guard</u> : there is a traffic direction command on the considered block. <u>Action</u> : Update IXL traffic commands.
IXL_command_overlap_locked	An IXL overlap locked command on a signal	<u>Guard</u> : there is no overlap locked command on the considered signal. <u>Action</u> : Update IXL traffic commands.
IXL_no_command_overlap_locked	Remove an IXL overlap locked command on a signal	<u>Guard</u> : there is an overlap locked command on the considered signal. <u>Action</u> : Update IXL overlap locked commands.

Relationship between notions

All booked blocks include booked blocks and overlap booked blocks. Traffic directions may be set or unset on booked blocks. Protected area blocks are established on booked blocks and overlap booked blocks when there is a permissive signal command. Blocks are still in protected areas when traffic directions are unset on them. When a block is un-booked, it is also removed from the protected area. The following figure illustrates these notions. Note that traffic directions are set on blocks from the signal S3 to S6, but the signal S3 has not changed to permissive aspect. Thus protected areas do not cover these blocks. Protected areas cover blocks between the signals S4 and S5 in spite of the fact that there is no traffic direction on these blocks.



Let B, B1 and B2 denote the set of all booked blocks, that of booked blocks and that of overlap booked blocks. Let T denote the set of blocks on which traffic directions are defined, and let P denote the set of blocks in protected areas. The following predicates and explications give an overview on the relations between these sets:

- $B = B1 \vee B2$
- $B1 \wedge B2 = \{\}$
- $T <: B1$
- $P <: B$
- $T \setminus P$: blocks on which traffic directions are set, but the corresponding signal is still in restrictive aspect;
- $P \setminus T$: overlap booked blocks or blocks on which traffic directions are unset but are still booked;
- $T \wedge P$: blocks on which traffic directions are set and that are reachable through a permissive aspect signal.

Blocks in all booked blocks are compatible and they are not blocks of points belonging to commanded points or conjugated points thereof.

3. Testing of the IXL-DC Event-B model

Our experience in development of formal models of software has shown that a model cannot be considered adequate or valid as long as the corresponding software has not been tested. There is no reason why the situation should be different at system level. This is why we planned in our case study to test the IXL-DC under realistic operating conditions.

With that purpose in mind, our initial intention was to develop manually a Classical-B model of the IXL-DC software starting from the IXL-DC Event-B model, then to develop manually the corresponding B0¹ model and the runtime environment (I/O procedures, scheduler, etc.), then to translate automatically this B0 model into an Ada program, then to generate automatically the corresponding executable code and finally to plug this code in Alstom’s Factory Integration & Validation Platform (FIVP) where signalling systems are tested under conditions close to real operating conditions.

The figure below illustrates the envisioned code generation process and architecture of the test environment.

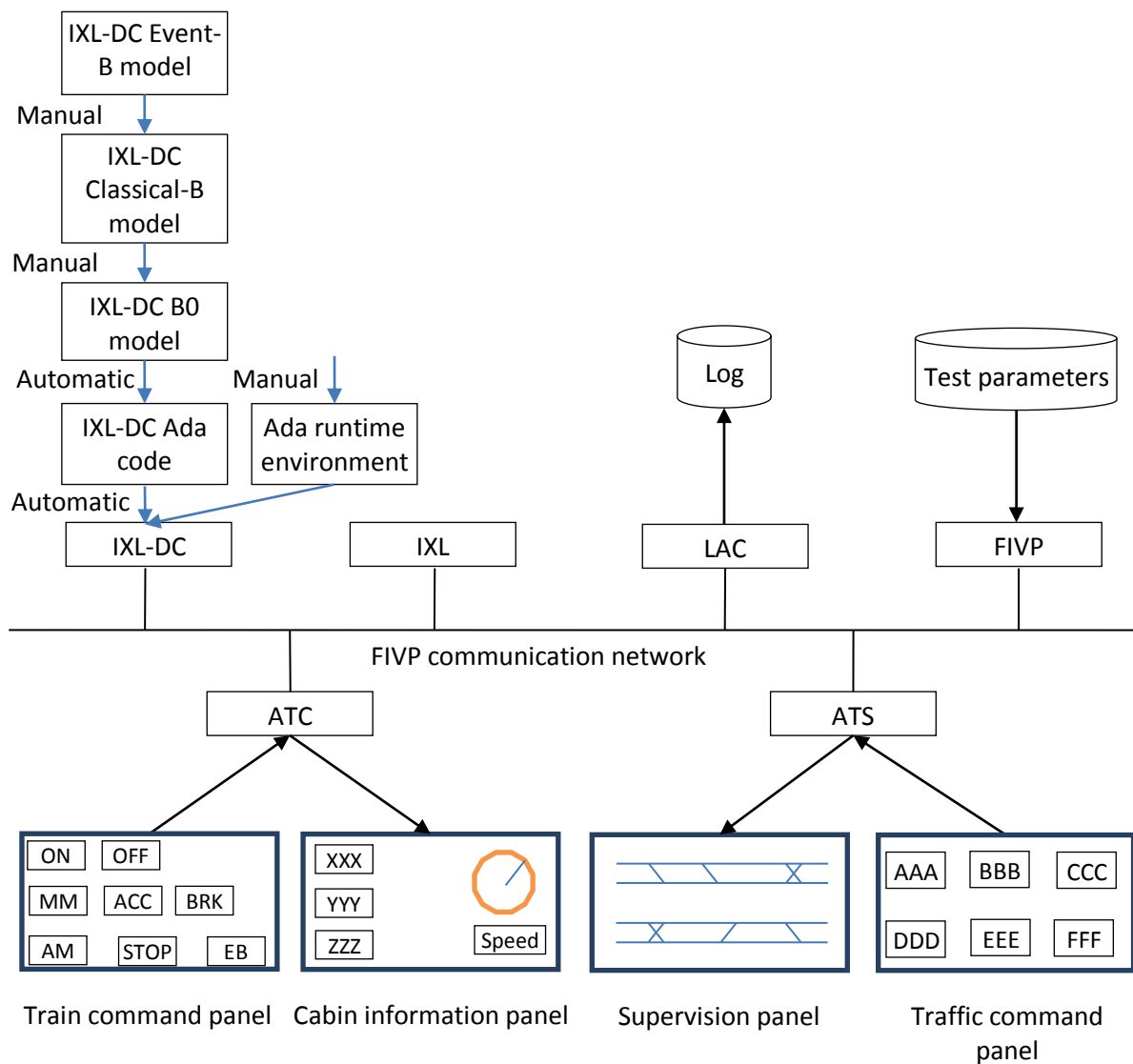


Figure 3.1. Envisioned IXL-DC code generation process and test environment architecture

¹ B0 is the subset of Classic-B that involves only the data types and instructions available in imperative programming languages like Ada or C.

In the previous figure, boxes ATS, ATC and IXL represent the actual sub-systems (hardware and embedded executable code); box IXL-DC represents only the executable code of IXL-DC; box LAC represents the Logic Communication Analyser, a probe that scrutinises all the messages exchanged between the components of the test environment and logs them in a file; and box FIVP represents the Factory Integration and Validation Platform mentioned above.

FIVP provides interfaces enabling the tester to play the Train Driver role (if the train control system is not fully automatic) and displays the cabin information according to the system state. FIVP simulates trains according to mathematical continuous models of kinetics and their physical characteristics of trains, and wayside devices (points, signals, track circuits, etc.).

We started to implement this process, but we realized after writing some Classic-B software components that it would take too long given the project deadlines and, in addition, that it would bring us no real methodological lesson given that we have long experience developing Classical-B models of software.

We therefore imagined a simpler and quicker to implement process.

First we abandoned the idea to plug the IXL-DC software in the FIVP because this requires non trivial developments. We decided instead to exploit the messages logged by the logic analyser of communications during tests as this log contains all the information required by the IXL-DC to control the IXL and is easily achievable.

Second, drawing inspiration from ProB and other interpreters of the B language, we created a module in Ocaml language providing a set of functions and iterators enabling straightforward translation of Event-B models in Ocaml language and generation of compiled executable code.

Third, in order to exploit test logs and visualise the IXL-DC behaviour, we created the Ocaml modules that schedule events, that read test logs and that visualise graphically the state of IXL-DC.

And fourth, we created data representing the track of the line used for testing. This test line is a real commercial line with 263 blocks, 112 track circuits, 40 points and 85 signals.

The figure below illustrates the implemented code generation process and the test environment of IXL-DC. This shows that the IXL-DC implementation can be run side-by-side with the standard test environment, driven by communications coming from the other components via the log

This test environment architecture allows realistic and reasonably efficient testing of the IXL-DC model. Indeed, the tested executable code of IXL-DC is close to the Event-B model; the test line is a real commercial line; the controlled IXL system is the actual IXL system operating with the actual ATS and ATC systems and with trains and wayside equipment modelled by mathematical continuous models developed with the MATLAB toolbox.

We have just begun testing with logs provided by the validation team of the signalling system developed for the test line used. The first positive results are that the run tests revealed gaps in data representing the line and that after some few adjustments they execute completely. That is to say that although the model has not been fully proved yet it is relatively sound (i.e. no unexpected abortions due, for instance, by division by 0). We shall now start the analysis of the tests results and this will lead certainly to modifications of the model.

We do not consider testing as a substitute for animation but rather as a complement. Animation is an essential step to develop models using representative but not necessarily industrial size cases and to ensure that their behaviour is effectively the envisioned behaviour. Animation permits an interactive analysis of properties of constants, variables and events. Testing on his side ensures that the behaviour of the model meets the needs for which the system was designed. Moreover, testing is done on an almost operating environment and deals with industrial size cases hardly compatible, at least with the current technology, with animation conditions.

Investigation: OCaml hand-translation compared to ProB

In a side experiment we have tried to perform the testing conducted using the OCaml hand-translated model using the ProB tool to run the original model, but reading in the log-data using

newly developed external I/O functions (external functions were added previously to ProB, and allow to link external Prolog code to B models). The results were very encouraging: we managed to replay the original log files about 2.5 times faster than the hand-translated OCaml code. We will analyse whether it is feasible to integrate ProB directly into the test-environment architecture (skipping the hand-translation process).

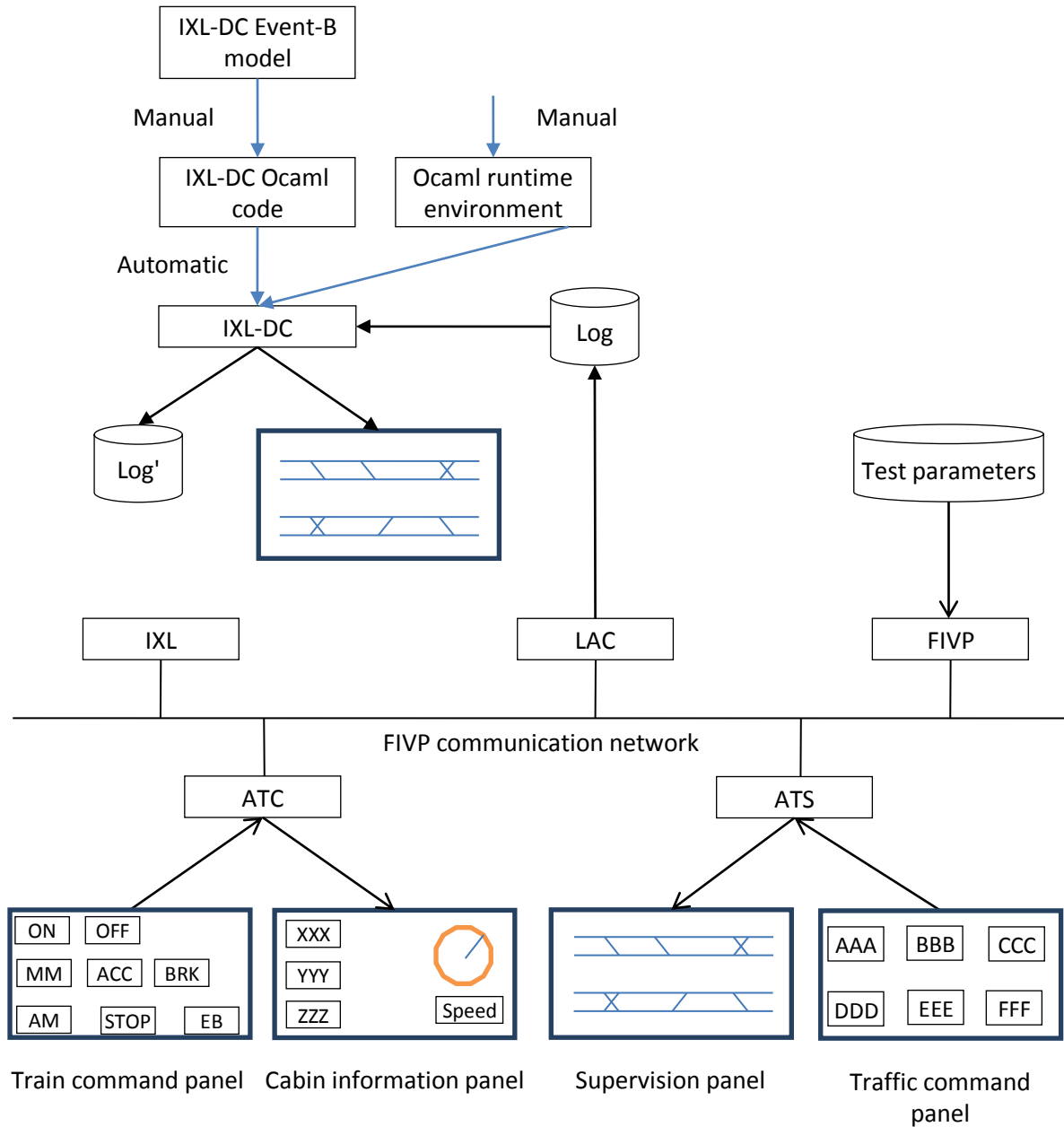


Figure 3.2. Implemented IXL-DC code generation process and test environment architecture

4. Intermediate assessment of ADVANCE tools

Within the first period of ADVANCE we defined requirements on the IXL-DC, created an Event-B model of it, proved some of its parts and validated its behaviour relatively to the behaviour we had in mind. We had some difficulties using the Theory plug-in of Rodin to prove the theories involved in the model and using ProB to animate the Event-B model involving theories. To overcome these problems we translated the Event-B model into Classical-B and animated that model with ProB. We reported these activities and difficulties in deliverable D1.2 ([6]).

During the second period of ADVANCE we concentrate on two activities. First, we undertook the exhaustive formal verification, through proof and model-checking, of the Event-B model of IXL-DC. For proof we used the refinement technique, Rodin's theorem provers and the improved Theory plug-in; for model checking we used ProB. Second, we undertook the validation of the model of IXL-DC using system tests involving indirectly an actual signalling system and a continuous model of a train.

In this section we present an assessment of the methods and tools provided by ADVANCE that we have used so far in our case study and on the tools that we would like ADVANCE to provide. This assessment is obviously not definitive since the methods and tools will evolve as a result of on-going improvements and of the needs identified by the cases studies.

We start assessing Event-B and the modelling approach based on refinement regarding its usability for modelling industrial sized systems.

4.1 Event-B

The Event-B language has two qualities: simplicity and "openness". For simplicity we mean that Event-B is based on a reduced set of simple concepts which facilitates learning and practice by non-formal methods experts and decreases the cost of formal proof. And for openness we mean that Event-B is not determined once for all, the user is able to extend the language according to its needs as long as it complies with some fundamental rules. Moreover, the development method based on refinement allows progressive enrichment of the model and facilitates proof.

However, these qualities create obstacles in an industrial environment if users do not find or are not able to develop easily the concepts that they needs for their particular application. Typically, regular "classic" formal methods practitioners, which are good candidates to use Event-B, might be confused by the absence of some basic data operators and structures (transitive closures, sequences), substitutions (IF THEN ELSE) and component composition/decomposition primitives (INCLUDES). It is therefore paramount that the basic platform provides a set of libraries and plug-in sufficiently rich to overcome this drawback.

A similar situation occurs with refinement. Refinement is one of the most effective techniques to reduce the number and complexity of proof obligations during the construction of Event-B models. However, the creation of a too large number of refinements renders the maintenance of the complete model impractical in an industrial context because the modification of an intermediate refinement can have a significant impact on the previous and subsequent refinements. The difficulty in this case is that no tool can assist the user in choosing the appropriate refinements; only expertise on refinement allows avoiding this pitfall. It is therefore paramount that the platform provides adequate user manuals and guides of refinement.

4.2 Rodin

Rodin is the platform supporting the construction, proof and validation of Event-B models. Rodin is based on an "open" architecture and is developed according to the "open" model on the principle of voluntary. Roughly speaking, Rodin is made of two parts: a core part providing all the necessary tools to construct, prove and maintain the consistency of raw Event-B models; and of a set of plug-ins extending the language or providing additional tools like text editors, model animators and requirements managers. The core part is developed and maintained essentially by Systemel and the plug-ins are essentially developed by academia (principally Southampton and Düsseldorf) or by regular users.

Unquestionably, the 'open' development model encourages innovation and apportions costs of development and maintenance of Rodin between several entities but, at the same time, it may be an obstacle to the adoption of the platform in the industry. The difficulties encountered during the first period of the project illustrate some dysfunctions that may compromise the industrial use of Rodin:

- Difficulty for some partners to commit on development or maintenance delays as they do not have stable development teams.

- Lack of synchronization between development teams makes that incompatible tools coexist in the same release of the platform.
- Absence of contractual guarantees.

ADVANCE must find a development model with the advantages of the "open" development model while giving to industry the guarantees of quality, predictability and sustainability it expects from methods and tool providers.

4.2.1 Theory plug-in

In the beginning of the second period, the Theory plug-in was quite difficult to use and had a lot of bugs that needed to be worked around. These bugs have been fixed since and the tools have become more and more usable.

There are nevertheless two annoyances left that should be fixed to bring the tool to industrial level:

1. When renaming a project, the proof file .bpr does not take into account the name changing. Status of proof obligations is not impacted by the project name changing, but references of proof obligations (in .bpr file) become incorrect. To overcome this inconvenience, one must modify manually the references to the project name in the .bpr file.

2. Most of the proofs that use rules provided by the Theory plug-in must be performed manually, although they could be automated if the Theory plug-in was supporting a mechanism of "trigger" for automatic rules.

Such a mechanism would apply automatically a rule only when some hypothesis is present. It would allow automating some rules that are currently manual, thus increasing proof productivity.

Moreover, such a mechanism would allow propagating sub-expressions matched in a required hypothesis to a sub-goal that is produced by the rule. For instance, one could write rules such as the following one (which is envisioned to be applied in backward style, where the starred antecedent is a required hypothesis):

$$\frac{(f : A \rightarrow B)^* \ \& \ x : A}{x : \text{dom}(f)}$$

4.3 ProB

During the first period of the project we used ProB to ensure, by interactive animation of a signalling system model involving the IXL-DC that the behaviour of the latter corresponds with the behaviour we had in mind. Due to the problems reported in the deliverable D1.2, we didn't use the ProB plug-in of Rodin which accepts Event-B models but instead its standalone Tcl/Tk version which accepts Classical-B models. This didn't change fundamentally our experience with ProB, except that we couldn't use BMotion Studio, the graphical editor integrated with Rodin and ProB that enables developers to create easily graphical visualization of animations, but the less user-friendly facility of the Tcl/Tk version of ProB enabling graphical animation.

During the second period of the project we used ProB to model-check the signalling system model involving the IXL-DC and to verify properties of the data describing the test line used in tests of the IXL-DC model (cf. section §3).

In all these uses ProB was quite effective. Graphical animation helped us to clarify and develop the behaviour of the IXL-DC. Model-checking, although it did not analyse the complete space, disclosed an erroneous traffic direction release done by the zone controller model. Data verification disclosed errors in the relation between blocks (virtual segments of track) and underlying track circuits (physical portions of track devoted to train detection) and in the relation between signals and turnaround blocks (segments of track where trains can change travel direction).

It would be worth however improving the user-unfriendly interactive B-expressions interpreter and providing an interface enabling to call ProB (or at least its B-interpreter) from external programs. We hope that this be done in ProB 2.0, which is being developed within the Advance project.

Also, we were very pleased with the responsiveness of the development team of ProB which developed or corrected the tools we needed to analyse animations. Moreover, the fact that a corporation, FormalMind, was created to develop and maintain ProB increases industrial confidence on the sustainability of this tool.

4.4 Co-simulation

Although we haven't used the co-simulation tools provided by ADVANCE we exploit co-simulation indirectly. Indeed, the FIVP used for testing Alstom's signalling systems and that produces the logs we use for testing the IXL-DC model simulates trains on the basis of a continuous mathematical model of kinetics and of trains' physical characteristics. Co-simulation is used also to predict and analyse the performance of a signalling system for a particular line in terms of maximum number of trains in operation, minimum travel time, minimum time between two trains, energy consumption and so on. The co-simulation platform involves the simulator of trains of FIVP and the MATLAB model several components of the signalling system.

That is to say that co-simulation is an important topic for Alstom and that it is of great interest that ADVANCE provides a co-simulation platform allowing to plug components developed with different formalisms : Event-B, Classic-B, Java, C and so on.

5. Contribution of ADVANCE Methods and Tools to Alstom's certification process

5.1 Certification framework

According to the French Standardization Organization (AFNOR) "Certification is a business process through which a recognized body acting independently with no ties to the parties involved gives written assurance that an organization, a process, a service, a product or a set of professional skills meets the baseline requirements set out in a reference standard."

Certification of a complete railway signalling system involves certification of many of the above-mentioned entities regarding requirements defined in many reference standards. In this section we shall concentrate on the certification of Alstom's safety critical railway systems regarding requirements defined in the CENELEC (European Committee for Electrotechnical Standardization) standards EN 50126 [1], EN 50128 [2] and EN 50129 [3] applicable to railway systems.

Thus, in our framework, certification is the process of achieving and obtaining acceptance of the proof of safety assurance through the activities of approval, assessment and cross acceptance.

To point out the certification activities prescribed by CENELEC standards to which ADVANCE methods and tools contribute we present in the following sections standards EN 50126 and EN 50129. We do not present standard EN 50128, although software assessment is a very important part of certification activities, because ADVANCE methods concern essentially system activities.

Almost all the text of this section is taken either from the standard themselves or from deliverable D.1.1 of OPENCROSS project [4].

5.1.1 CENELEC standard EN 50126

This standard provides a process which enables the implementation of a consistent approach to the management of reliability, availability, maintainability and safety (denoted by the acronym RAMS) of total railway systems (including but not restricted to signalling). This Standard:

- Defines RAMS in terms of reliability, availability, maintainability and safety and their interaction;
- Defines a process, based on the system life-cycle and tasks within it, for managing RAMS;

- Enables conflicts between RAMS elements to be controlled and managed effectively;
- Defines a systematic process for specifying requirements for RAMS and demonstrating that these requirements are achieved.

The system life-cycle defined in EN 50126 involves fourteen phases and for each of them the Standard defines the objectives, the inputs, the requirements on the results and the deliverables regarding RAMS.

The system life-cycle phases and the related safety tasks are presented in the table below:

Life-cycle phase	Phase related safety tasks
1. Concept	<ul style="list-style-type: none"> • Review Previously Achieved Safety Performance • Consider Safety Implications of Project • Review Safety Policy & Safety Targets
2. System definition and application conditions	<ul style="list-style-type: none"> • Evaluate Past Experience Data for Safety • Perform Preliminary Hazard Analysis • Establish Safety Plan (Overall) • Define Tolerability of Risk Criteria • Identify Influence on Safety of Existing Infrastructure Constraints
3. Risk analysis	<ul style="list-style-type: none"> • Perform System Hazard & Safety Risk Analysis • Set-Up Hazard Log • Perform Risk Assessment
4. System requirements	<ul style="list-style-type: none"> • Specify System Safety Requirements (Overall) • Define Safety Acceptance Criteria (Overall) • Define Safety Related Functional Requirements • Establish Safety Management
5. Apportionment of system requirements	<ul style="list-style-type: none"> • Apportion System Safety Targets & Requirements <ul style="list-style-type: none"> - Specify Sub-System & Component Safety Requirements - Define Sub-System & Component Safety Acceptance Criteria • Update System Safety Plan
6. Design and implementation	<p>Implement Safety Plan by Review, Analysis, Testing and Data Assessment, addressing:</p> <ul style="list-style-type: none"> • Hazard Log • Hazard Analysis & Risk Assessment • Justify Safety Related Design Decisions • Undertake Programme Control, covering: <ul style="list-style-type: none"> - Safety Management - Control of Sub-Contractors & Suppliers • Prepare Generic Safety Case • Prepare (if appropriate) Generic Application Safety Case
7. Manufacturing	<ul style="list-style-type: none"> • Implement Safety Plan by: Review, Analysis, Testing & Data Assessment • Use Hazard Log

Life-cycle phase	Phase related safety tasks
8. Installation	<ul style="list-style-type: none"> • Establish Installation Programme • Implement Installation Programme
9. System validation (including safety acceptance and commissioning)	<ul style="list-style-type: none"> • Establish Commissioning Programme • Implement Commissioning Programme • Prepare Application Specific Safety Case
10. System acceptance	<ul style="list-style-type: none"> • Assess Application Specific Safety Case
11. Operation and maintenance	<ul style="list-style-type: none"> • Undertake On Going Safety Centred Maintenance • Perform On Going Safety Performance Monitoring and Hazard Log Maintenance
12. Performance monitoring	<ul style="list-style-type: none"> • Collect, Analyse, Evaluate and Use Performance & Safety Statistics
13. Modification and retrofit	<ul style="list-style-type: none"> • Consider Safety Implications for Modification & Retrofit
14. Decommissioning and disposal	<ul style="list-style-type: none"> • Establish Safety Plan • Perform Hazard Analysis & Risk Assessment • Implement Safety Plan

Table 5.1. Safety Tasks for each phase defined in EN50126

5.1.2 CENELEC standard EN 50129

This standard provides a set of requirements that shall be satisfied in order that a safety critical railway system/sub-system/equipment can be accepted as adequately safe for its envisioned application. The requirements are presented under three headings:

- Evidence of quality management;
- Evidence of safety management;
- Evidence of functional and technical safety.

All of these conditions shall be satisfied, at equipment, sub-system and system levels, before the safety related system can be accepted as adequately safe.

The documentary evidence that these conditions have been satisfied shall be included in a structured safety justification document, known as the Safety Case. The Safety Case forms part of the overall documentary evidence to be submitted to the relevant safety authority in order to obtain safety approval for a generic product, a class of application or a specific application.

The Safety Case contains the documented safety evidence for the system/sub-system/equipment, and shall be structured as follows:

Part 1 Definition of System (or sub-system/equipment): This shall precisely define or reference the system/sub-system/equipment to which the Safety Case refers, including version numbers and modification status of all requirements, design and application documentation.

Part 2 Quality Management Report: This shall contain the evidence of quality management, as specified in 5.2 of this standard.

Part 3 Safety Management Report: This shall contain the evidence of safety management, as specified in 5.3 of this standard.

Part 4 Technical Safety Report: This shall contain the evidence of functional and technical Safety, as specified in 5.4 of this standard.

Part 5 Related Safety Cases: This shall contain references to the Safety Cases of any sub-systems or equipment on which the main Safety Case depends.

It shall also demonstrate that all the safety-related application conditions specified in each of the related sub-system/equipment Safety Cases are either fulfilled in the main Safety Case, or carried forward into the safety-related application conditions of the main Safety Case.

Part 6 Conclusion: This shall summarise the evidence presented in the previous parts of the Safety Case, and argue that the relevant system/sub-system/equipment is adequately safe, subject to compliance with the specified application conditions.

5.1.2.1 Evidence of quality management

The first condition for safety acceptance that shall be satisfied is that the quality of the system, sub-system or equipment has been, and shall continue to be, controlled by an effective quality management system throughout its life-cycle. Documentary evidence to demonstrate this shall be provided in the Quality Management Report, which forms Part 2 of the Safety Case.

5.1.2.2 Evidence of safety management

The second condition for safety acceptance which shall be satisfied is that the safety of the system, sub-system or equipment has been, and shall continue to be, managed by means of an effective safety management process, which should be consistent with the management process for RAMS described in EN 50126. The purpose of this process is to further reduce the incidence of safety-related human errors throughout the life-cycle, and thus minimise the residual risk of safety-related systematic faults.

5.1.2.2.1 Safety life-cycle

The safety management process consists of a number of phases and activities, which are linked to form a safety life-cycle consistent with the system life-cycle defined in EN 50126. The life-cycle as modelled is a typical development V cycle, with the requirement specification, architecture and detailed design and development phases as top-down phases and the integration and validation phases as bottom-up phases.

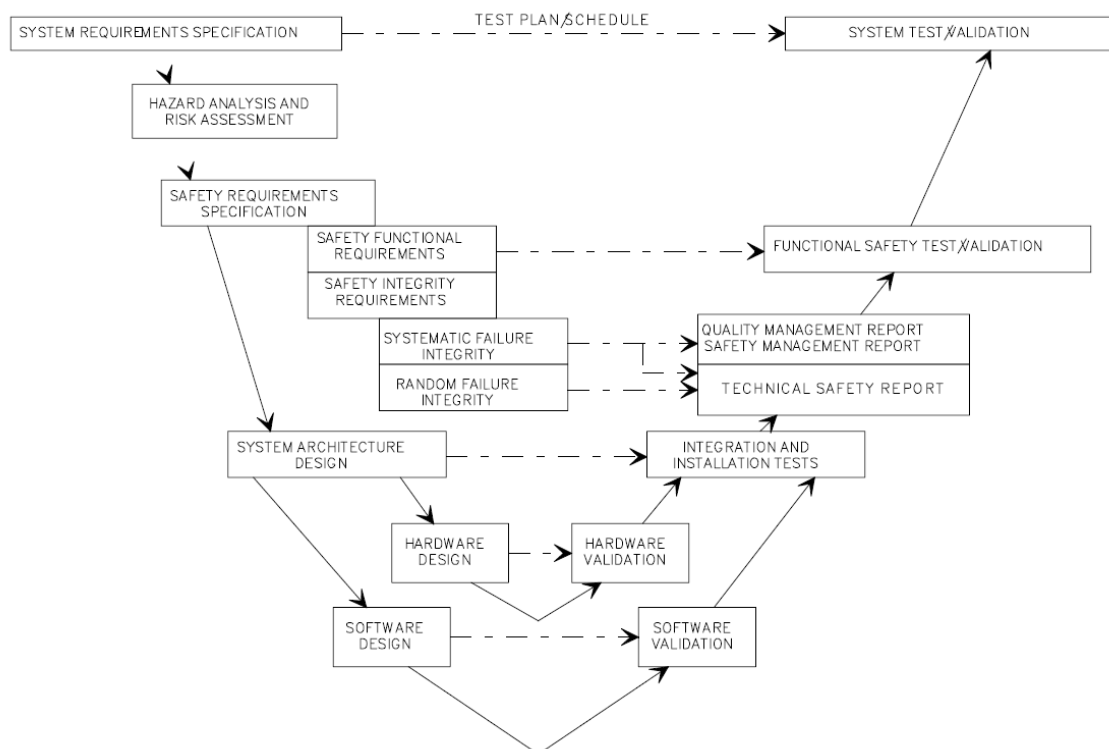


Figure 5.2. Example of design and validation life-cycle defined in EN50129

5.1.2.2.2 Safety plan

A Safety Plan shall be drawn up at the start of the life-cycle. This plan shall identify the safety management structure, safety-related activities and approval mile-stones throughout the life-cycle, and shall include the requirements for review of the Safety Plan at appropriate intervals. The Safety Plan shall be updated and reviewed if subsequent alterations or additions are made to the original system/sub-system/equipment. If any such change is made, the effect on safety shall be assessed, starting at the appropriate point in the life-cycle.

The Safety Plan shall deal with all aspects of the system/sub-system/equipment, including both hardware and software. EN 50128 shall be referenced for Software aspects.

The Safety Plan should include a Safety Case Plan, which identifies the envisioned structure and principal components of the final Safety Case.

5.1.2.2.3 Hazard log

A Hazard Log shall be created and maintained throughout the safety life-cycle, as explained in EN 50126.

It shall include a list of identified hazards, together with associated risk classification and risk control information for each hazard. The Hazard Log shall be updated if any modification or alteration is made to the system, sub-system or equipment.

5.1.2.2.4 Safety requirements specification

The specific safety requirements for each system/sub-system/equipment, including safety functions and safety integrity, shall be identified and documented in the Safety Requirements Specification. This shall be achieved by means of - Hazard Identification and Analysis,

- Risk Assessment and Classification,
- allocation of Safety Integrity Levels,

as explained in EN 50126. Some information concerning Safety Integrity Levels for railway electronic systems is contained in Annex A.

NOTE The Safety Requirements Specification may be included in the system/sub-system/equipment Functional Requirements Specification or may be written as a separate document.

5.1.2.2.5 System/sub-system/equipment design

This phase of the life-cycle shall create a design which fulfils the specified operational and safety requirements. A top-down, structured design methodology shall be used, with rigorously controlled and reviewed documentation. In particular, the relationship between hardware and software, as represented by the Software Requirements Specification and software/hardware integration, shall be strictly managed, and the standard EN 50128 shall be adhered to.

5.1.2.2.6 Safety reviews

Safety reviews shall be carried out at appropriate stages in the life-cycle. Such reviews shall be specified in the Safety Plan, and their results fully documented. Any alteration or extension to the system, sub-system or equipment shall also be subject to review.

5.1.2.2.7 Safety verification and validation

The Safety Plan shall include or reference plans for verifying that each phase of the life-cycle satisfies the specific safety requirements identified in the previous phase, and for validating the completed system/sub-system/equipment against its original Safety Requirements Specification.

These activities shall be carried out and fully documented, including appropriate testing and safety analyses. They shall be repeated as appropriate in the event of any subsequent modification or addition to the system/sub-system/equipment.

The degree of independence necessary for the verifier and the validator shall be in accordance with the Safety Integrity Level of the system/sub-system/equipment under scrutiny.

At the discretion of the safety authority, the assessor may be part of the supplier's organisation or of the customer's organisation but, in such cases, the assessor shall

- be authorised by the safety authority,
- be totally independent from the project team,
- report directly to the safety authority.

5.1.2.2.8 Safety justification

The evidence that the system/sub-system/equipment meets the defined conditions for safety acceptance shall be presented in a structured safety justification document known as the Safety Case, as explained in 5.1 of this standard.

5.1.2.2.9 System/sub-system/equipment handover

Prior to handover of the system/sub-system/equipment to a railway authority, the conditions for safety acceptance and safety approval defined in 5.5 shall be satisfied, including submission of the Safety Case and the Safety Assessment Report.

5.1.2.2.10 Operation and maintenance

Following handover, the procedures, support systems and safety monitoring defined in the Safety Plan and in Section 5 of the Technical Safety Report (part of the Safety Case) shall be adhered to. During the operational life of a system, change requests may be raised for a variety of reasons, not all of which will be safety-related. Each change request shall be assessed for its impact on safety, by reference to the relevant portion of the safety documentation. Where a change request results in a modification which could affect the safety of the system, or associated systems, or the environment, the appropriate portion of the safety life-cycle shall be repeated to ensure that the implemented modification does not unacceptably reduce the level of safety. Table E.10 gives guidance on Application, Operation and Maintenance for each Safety Integrity Level.

5.1.2.2.11 Decommissioning and disposal

At the end of the operational life of a system, its decommissioning and disposal shall be carried out in accordance with the measures defined in the Safety Plan and in Section 5 of the Technical Safety Report (part of the Safety Case).

5.1.2.3 Evidence of functional and technical safety

In addition to the evidence of quality and safety management, there is a third condition that must be satisfied before a system/sub-system/equipment can be accepted as adequately safe for its envisioned application. This consists of technical evidence for the safety of the design, which shall be documented in the Technical Safety Report.

The Technical Safety Report shall be arranged under the following headings:

Section 1 Introduction: This section shall provide an overview description of the design, including a summary of the technical safety principles that are relied on for safety and the extent to which the system/sub-system/equipment is claimed to be safe in accordance with this standard.

Section 2 Assurance of correct functional operation: This section shall contain all the evidence necessary to demonstrate correct operation of the system/sub-system/ equipment under fault-

free normal conditions (that is, with no faults in existence), in accordance with the specified operational and safety requirements.

The following aspects shall be included:

- 2.1 System architecture description;
- 2.2 Definition of interfaces;
- 2.3 Fulfilment of System Requirements Specification;
- 2.4 Fulfilment of Safety Requirements Specification;
- 2.5 Assurance of correct hardware functionality;
- 2.6 Assurance of correct software functionality.

Section 3 Effects of faults: This section shall demonstrate that the system/sub-system/equipment continues to meet its specified safety requirements, including the quantified safety target, in the event of random hardware faults. In addition, a systematic fault could still exist, despite the quality and safety management processes defined in 5.2 and 5.3 of this standard. This section shall demonstrate which technical measures have been taken to reduce the consequent risk to an acceptable level.

This section shall also include demonstration that faults in any system/sub-system/equipment having a Safety Integrity Level lower than that of the overall system, including Level 0, cannot reduce the safety of the overall system.

The following headings shall be used in this section.

- 3.1 Effects of single faults;
- 3.2 Independence of items;
- 3.3 Detection of single faults;
- 3.4 Action following detection (including retention of safe state);
- 3.5 Effects of multiple faults;
- 3.6 Defence against systematic faults.

Section 4 Operation with external influences: This section shall demonstrate that when subjected to the external influences defined in the System Requirements Specification, the system/sub-system/equipment - continues to fulfil its specified operational requirements,- continues to fulfil its specified safety requirements (including fault conditions).

The Safety Case is therefore valid only within the specified range of external influences, as defined in the System Requirements Specification. Safety is not assured outside these limits, unless additional special measures are provided.

The methods used to withstand the specified external influences shall be fully explained and justified.

Section 5 Safety-related application conditions: This section shall specify (or reference) the rules, conditions and constraints which shall be observed in the application of the system/sub-system/equipment. This shall include the application conditions contained in the Safety Case of any related sub-system or equipment.

Section 6 Safety Qualification Tests: This section shall contain evidence to demonstrate successful completion, under operational conditions, of the Safety Qualification Tests.

5.2 Introduction of ADVANCE Methods and Tools in Alstom's system development process

The figure below illustrates the system development process of Alstom and the activities related to safety (in red) and to verification and validation (in green). This process has been assessed compliant with CENELEC EN 50129 standard by independent safety assessors and our intention is to insert in it the activities, methods and tools developed in the ADVANCE project without disrupting its structure and balance.

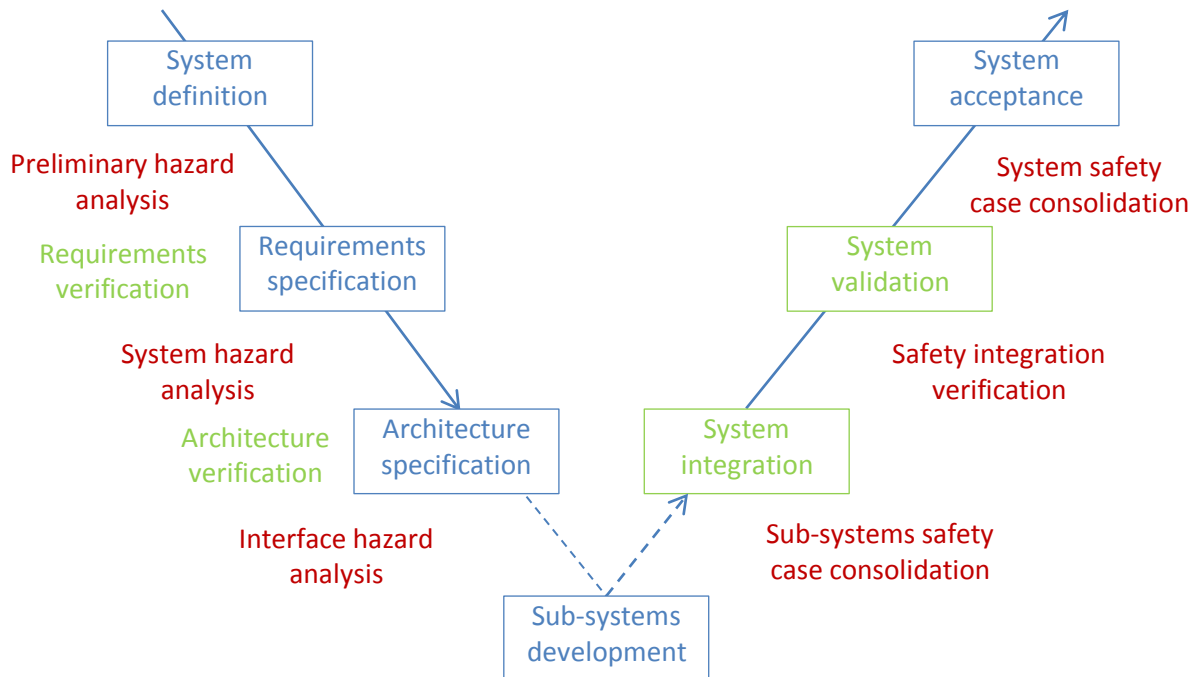


Figure 5.3. Alstom's system development process compliant with EN50129

In the sequel we present the contribution of ADVANCE to the achievement of the activities of each of the phases of the Alstom process.

5.2.1 System Definition

The purpose of this phase is to:

- Establish system mission profile,
- Prepare system description,
- Identify operation and maintenance strategies,
- Identify operating conditions,
- Identify maintenance conditions,
- Identify Influence of existing infrastructure constraints.

5.2.1.1 Contribution of ADVANCE

None of the activities, methods and tools of ADVANCE contributes to the achievement of the previous tasks.

5.2.2 Preliminary Hazard Analysis

Starting from the operational context/procedures and the defined system, this analysis aims at:

- identifying the hazards at the boundary of the system under consideration (resulting from the use of the system in a specified context, as defined in the system definition);
- classifying the consequences (possible accidents) of these hazards;

- identifying the necessary mitigations for the system or the elements of the system (including preliminary SIL allocation to functions), in order to lower the risk to an acceptable level;
- (optional) predict Hazardous Failure Rate (HFR) achievable (to be confirmed by Fault Tree Analysis or equivalent method after detailed design).

5.2.2.1 Contribution of ADVANCE

ADVANCE promotes the use of STAMP and STPA for safety analyses. According to Thomas and Leveson [5], STAMP (System's Theoretic Accident and Models) is a model of accident causation that treats safety as a control problem, rather than as a failure problem.

STAMP is based on the observation that there are four types of hazardous control actions that need to be eliminated or controlled to prevent accidents:

- 1) A control action required for safety is not provided or is not followed
- 2) An unsafe control action is provided that leads to a hazard
- 3) A potentially safe control action is provided too late, too early, or out of sequence
- 4) A safe control action is stopped too soon or applied too long

STPA (System's Theoretic Process Analysis) is a hazard analysis technique built on STAMP. Identifying the potentially unsafe control actions for the specific system being considered is the first step in STPA. These unsafe control actions are used to create safety requirements and constraints on the behaviour of both the system and its components. Additional analysis can then be performed to identify the detailed scenarios leading to the violation of the safety constraints. As in any hazard analysis, these scenarios are then used to control or mitigate the hazards in the system design.

Railway industry has a long experience using conventional, well known, safety analysis techniques: Fault Trees Analysis, Failure Mode Effect Analysis, etc. and, indeed, these are the only safety techniques recommended in CENELEC standards. Therefore, it seems to us unrealistic to propose to introduce STAMP and STPA without a very deep and careful examination of their pros and cons. For the time being ADVANCE contributes to our introduction to these techniques, to dissemination of this knowledge in Alstom and to encourage Alstom to go further in their evaluation.

5.2.3 Requirements Specification

The requirements specification phase aims at:

- Undertake requirements analysis
- Specify system
- Specify environment
- Define system demonstration and acceptance criteria
- Establish validation plan
- Establish management, quality and organisation requirements
- Implement change control procedure

5.2.3.1 Contribution of ADVANCE

ADVANCE contributes to achieve the Requirements specification phase by introducing a formal model development process made of three activities: modelling, simulation and proof.

Starting from the informal Requirements Specification document and from the Preliminary Hazard Analysis document the modelling activity shall consist in creating with the Rodin tool an Event-B model of critical parts of the system and in discharging the proof obligations ensuring that the model is sound or well defined (e.g. no division by 0). The Event-B model shall be verified by the verification and safety assurance teams regarding compliance with functional and safety requirements.

The simulation activity shall consist in animating with ProB and/or co-simulating with FMI the Event-B model created by the modelling activity. The simulation report shall describe the animation/co-simulation scenarios: their objectives, their steps and the expected behaviour of the model. The

relevance of the scenarios regarding the envisioned behaviour of the system shall be verified by the verification team. The animation report shall be used by the validation team for the definition of system validation tests.

The proof activity shall consist in discharging the proof obligations of the Event-B model generated by Rodin. The proof report shall be verified by the verification team regarding completeness and correctness of the proof.

The figure below illustrates the formal development in the Requirements Specification phase.

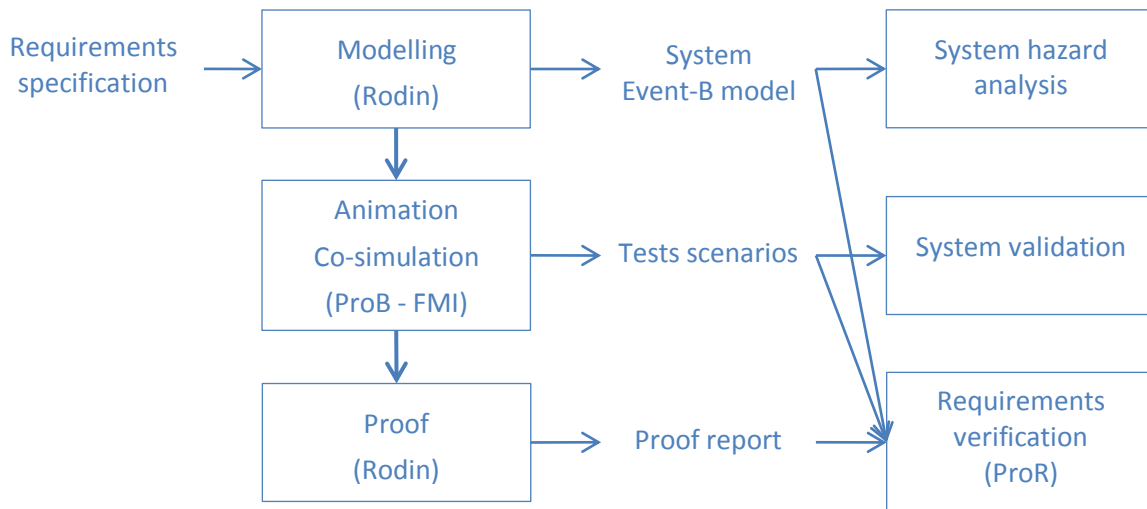


Figure 5.4. Requirements Specification formal model development process

5.2.4 Requirements Verification

The requirements verification activity aims at verifying completeness and consistency of system requirements and external interface description.

5.2.4.1 Contribution of ADVANCE

Currently, completeness verification consists of ensuring that system requirements capture all system needs identified in the previous step; and consistency verification consists of ensuring that requirements define sound, complementary and adequate functions. To achieve completeness verification verifiers construct a traceability matrix using standard requirements management tools (Doors, Reqtify, etc.). ADVANCE will leave this unchanged. To achieve consistency verification verifiers undertake document reviews. ADVANCE will provide rigorous methods and tools to achieve this task.

There will be three verifications related to formal development: the Event-B model verification, the simulation verification and the proof verification.

The Event-B model verification ensures that the model formalises adequately the concepts and requirements of the system and therefore that it is a sound basis for further formal development. The effectiveness of the verification will be improved because the formal model is rigorous and therefore reduces the ambiguities of interpretation. Furthermore, the ProR tool shall help verifiers keep track of requirements in the Event-B model and to generate the traceability matrix between system requirements and formulae in the Event-B model. The Event-B model verification report shall be used for consolidation of the System Safety Case.

The simulation verification ensures that the simulation scenarios are relevant and that they cover all the features of the Event-B model. Model simulation and simulation verification are new activities that will improve considerably the confidence on the system being developed. The simulation verification report shall be used for consolidation of the System Safety Case.

The proof verification ensures completeness and correctness of proofs of proof obligations of the Event-B model generated by Rodin. Proof obligations represent the sufficient conditions for the model to be consistent. Thus, the verification of completeness and correctness of proofs of proof obligations shall ensure consistency of the Event-B model and contribute to check consistency of requirements specification. The proof verification report shall contain evidence that the proof rules added by the user, if any, are valid mathematical lemmas. The proof verification report shall be used for consolidation of the System Safety Case.

5.2.5 System Hazard Analysis

This analysis aims at:

- identifying the cause and consequences of the failures of the functions and interfaces supported by the system on the basis of system requirements specification and external interfaces description;
- identifying the mitigations necessary to control the hazards and to lower the risk at an acceptable level;
- confirming the SIL allocation to the functions and interfaces of the system,
- recording the hazards identified with their effects and the associated risk mitigation recommendations.

5.2.5.1 Contribution of ADVANCE

Safety Hazard Analysis shall identify the safety requirements captured in the Event-B model, verify that they are correctly formalised and log that information in the Hazard Log (cf. 5.1.2.2.3).

The motivation for doing this is that if finally the formalisation of a system safety requirement is present in the model of the formally developed software of one its sub-systems it will be possible to close the related hazard in the Hazard Log through mathematical proof rather than through testing. This shall be definitively an improvement in terms of safety evidence and furthermore shall reduce the number of tests related to safety.

5.2.6 Architecture Specification

This activity aims at:

- Apportion System Requirements
- Specify sub-systems and component requirements
- Define sub-systems and component acceptance criteria

5.2.6.1 Contribution of ADVANCE

ADVANCE contributes to achieve this activity by introducing a refinement process involving two activities: refinement/decomposition of the system Event-B model and proof.

The refinement/decomposition activity shall consist in refining the properties and events of the Event-B model of the system in order to apportion them between the sub-systems of the system. The refinement technique of Event-B and the composition/decomposition plug-in of Rodin shall be used. This activity shall produce an Event-B model involving several refinements of the Event-B model of the system. This Event-B model shall be verified by the verification and safety assurance teams regarding respectively the adequacy and correctness of the models and the formalisation of safety requirements.

The proof activity shall consist in discharging the proof obligations of the Event-B model generated by Rodin. The proof report shall be verified by the verification team regarding completeness and correctness of the proof.

The figure bellow illustrates the formal development in the Architecture Specification phase/

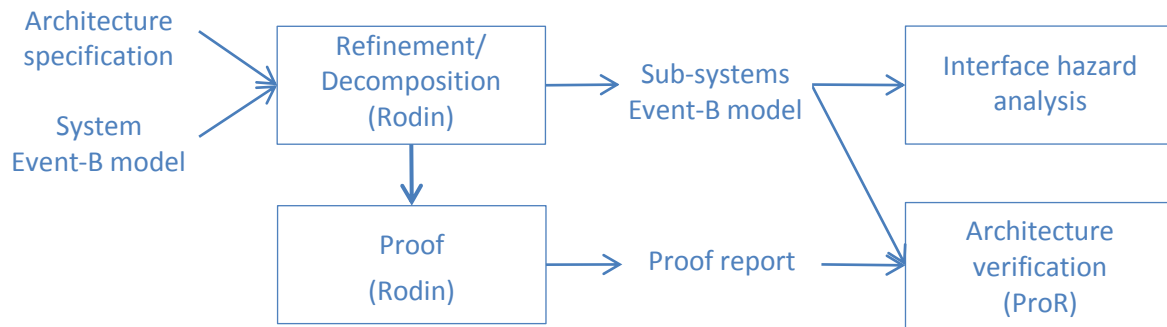


Figure 5.5. Architecture Specification formal model development process

5.2.7 Architecture verification

This activity aims at verify consistency and completeness of system architecture and internal interfaces descriptions.

5.2.7.1 Contribution of ADVANCE

The verification of the Event-B model of sub-systems shall contribute to check consistency of this model regarding architecture specification. The ProR tool shall assist verifiers to trace system and architecture requirements in the Event-B model and to generate the traceability matrix of the verification report. The Event-B model verification report shall be used for consolidation of the System Safety Case.

The proof obligations of the Event-B model of sub-systems generated by Rodin represent the necessary and sufficient conditions for this model to be consistent and compliant with the Event-B model of the system. Therefore, the verification of completeness and correctness of proofs of proof obligations shall ensure consistency of the Event-B model of sub-systems and compliance of this model with the Event-B model of the system. The proof verification report shall contain evidence that the proof rules added by the user, if any, are valid mathematical lemmas. The proof verification report shall be used for consolidation of the System Safety Case.

5.2.8 Interface Hazard analysis

This analysis aims at:

- identifying the cause and consequences of the failures of internal interfaces of the system on the basis of system architecture and internal interfaces description;
- identifying the mitigations necessary to control the hazards and to lower the risk at an acceptable level;
- confirming the SIL allocation to the components of the system,
- recording the hazards identified with their effects and the associated risk mitigation recommendations.

5.2.8.1 Contribution of ADVANCE

Interface Hazard Analysis shall identify the safety requirements captured in the Event-B model of sub-systems, verify that they are correctly formalised, refined and apportioned and log that information in the Hazard Log (cf. 5.1.2.2.3).

5.2.9 Sub-systems development

The development process of each sub-system depends on the nature of the sub-system.

If the sub-system itself is decomposed in sub-systems, like for instance the ATS and ATC sub-systems then, except minor details, the process described here shall be followed. If the sub-system is not

decomposed into sub-systems, like for instance the wayside zone controller, then the development of its hardware and the development of its software start in parallel.

5.2.9.1 Contribution of ADVANCE

When one of the sub-systems to be developed is not decomposed into sub-systems and is safety-critical its software is developed formally with Classical-B method. In this case, the Event-B model of the sub-system shall be used to create the initial Classical-B model of the software. This Classical-B model is not a refinement of the Event-B model, in the B sense of the term. Therefore, it will not be possible to prove formally compliance of the Classical-B model with the Event-B model. However, the former shall be generated systematically from the latter according to a validated predefined schema [6] that will ensure compliance. We shall have a continuous and homogeneous formal process from system development to software development. Ensuring the soundness of the translation is a topic for future research.

The figure below is an extract of the Software Development Work Method Statement of Alstom Transport Information Solutions. It illustrates the development process of safety-critical software with Classical-B.

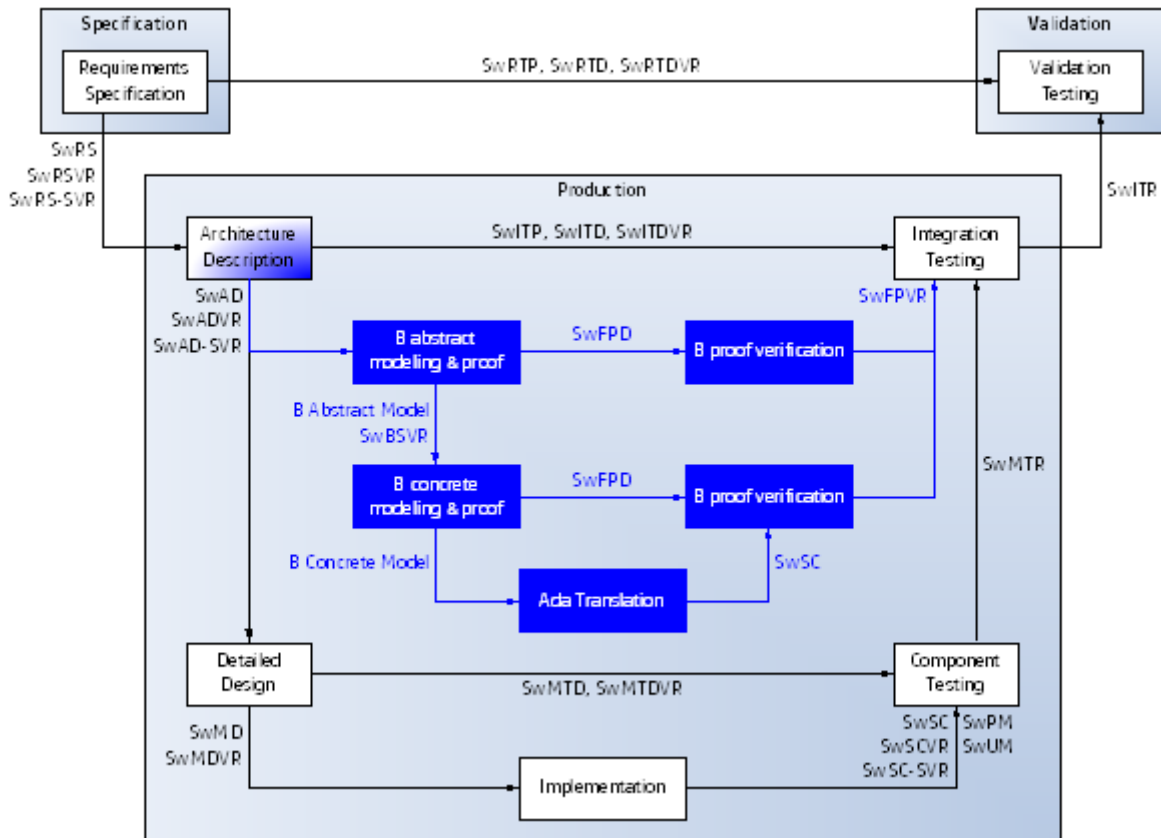


Figure 5.6. Alstom’s Software development process with Classical-B

It should be noted that in this process component testing is not performed on components developed with Classical-B and that integration testing is performed only to test interfaces between components developed with Classical-B and components not developed with Classical-B.

Likewise we shall take advantage of the use of Event-B for system development and of Classical-B for software development to reduce system testing activities.

5.2.10 Sub-system safety cases consolidation

This activity aims at:

- assess the Safety Cases of the sub-systems,

- identify and record the hazards closed by sub-system development,
- analyse the consequences on safety of the constraints exported by the sub-systems and, if necessary, integrate them into the Hazard Log.

5.2.10.1 Contribution of ADVANCE

ADVANCE contributes indirectly to this activity through safety analysis and verifications reports produced by the safety and verifications teams during the development of the sub-systems.

5.2.11 System integration

This activity aims at:

- assemble all the sub-systems,
- test compliance with system architecture and internal interface descriptions.

5.2.11.1 Contribution of ADVANCE

Requirements on the interface of a sub-system that have been formalised and proved in an Event-B model need not to be tested if the software of the sub-system has been formally developed and if its Classical-B model captures these requirements too.

5.2.12 Safety integration verification

This activity aims at verify that safety requirements closing hazards related to architecture and internal interfaces have been effectively satisfied either by testing or by formal proof of Event-B and Classical-B models.

5.2.12.1 Contribution of ADVANCE

ADVANCE contributes indirectly to this activity through safety analysis and verifications reports produced by the safety and verifications teams during the development of the sub-systems.

5.2.13 System validation

This activity aims at:

- perform tests ensuring fulfilment of system requirements,
- analyse system verification and integration reports,
- analyse sub-system verification and validation reports.

5.2.13.1 Contribution of ADVANCE

As for system integration, requirements on the system that have been formalised and proved in an Event-B model need not be tested if they have been refined and apportioned between sub-systems formally developed and proved.

5.2.14 System safety case consolidation

This activity aims at produce a System Safety Case compliant with CENELEC EN 50129 standard as described in §5.1.2.

5.2.14.1 Contribution of ADVANCE

5.2.15 System acceptance

This activity aims at:

- assess that the signalling system meets the customer requirements
- assess reliability, availability and maintainability demonstration,
- assess application specific System Safety Case

5.2.15.1 Contribution of ADVANCE

ADVANCE contributes indirectly to this activity through safety analysis and verifications reports produced by the safety and verifications teams during the development of the sub-systems.

6. Concluding

Considerable advances have been made in WP1 during Period 2 of the project and these are documented here. We have demonstrated the feasibility of a refinement based approach to modelling, simulation and verification of a complex interlocking system. We have identified both strengths and weaknesses of the existing Rodin tools for Event-B and found that considerable improvements have been made to the tools over the last period. Our experiences with the development of a refinement strategy approach will provide important guidance for other systems engineers in the future, both within Alstom and externally. The weaknesses we have identified in Rodin is helping the tool developers to prioritise there development plans. We have also developed a thorough understanding of where and how the ADVANCE methods and tools will contribute to enhancing the Alstom safety development process. Since the Alstom process id derived from processes defined by European standards, we believe many of these insights will be applicable to safety development in other domains.

References

- [1] CENELEC Standard EN 50126 : Railway applications — The specification and demonstration of Reliability Availability, Maintainability and Safety (RAMS); 1999.
- [2] CENELEC Standard EN 50128 : Railway applications — Communication, signalling and processing systems -Software for railway control and protection systems; October 2011.
- [3] CENELEC Standard EN 50129 : Railway applications — Communication, signalling and processing systems — Safety related electronic systems for signalling. February 2003.
- [4] Constraints of the certification process, D1.1, OpenCoss: Open Platform for the Evolutionary Certification of Safety-critical systems, 28 March 2012. EC 7th Framework Programme.
- [5] Performing Hazard Analysis on Complex, Software- and Human-Intensive Systems, J. Thomas, N. G. Leveson; Massachusetts Institute of Technology.
- [6] FP7 ADVANCE project, Proof of Concept Application in Railway Domain, D1.2 .