FP7 Project 2007- Grant agreement n°: 218575

Project Acronym: **INESS**

Project Title: **INtegrated European Signalling System**

Instrument: Large-scale integrating project
Thematic Priority: Transport

WS D – Generic Requirements

Deliverable D4.4_Safety Requirements in a Formal Format

Due date of deliverable	July 2010
Actual submission date	July 2010

Deliverable ID:	**D4.4**
Deliverable Title:	Safety Requirements in a Formal Format
WP related:	D.4
Responsible partner:	York, Southampton, LaQuSo (Eindhoven,Twente)
Task/Deliverable leader Name:	Osmar Marchi dos Santos (York)
Contributors:	York, Southampton, LaQuSo (Eindhoven,Twente), UIC, Railsafe

Start date of the project: 01-10-2008 Duration: 36 Months

Project coordinator: George Barbu
Project coordinator organisation: UIC

Revision: WS Final Dissemination Level[1]: CO

[1] PU: Public, PP: Restricted to other programme participants (including the Commission Services), RE: Restricted to a group specified by the consortium (including the Commission Services), CO: Confidential, only for members of the consortium (including the Commission Services).

Document Information

Document type: Report
Document Name: INESS_WS D_Deliverable D4.4_Safety Requirements in a Formal Format
Revision: WS Final
Revision Date: 26-10-2010
Author: York, Southampton, LaQuSo (Eindhoven,Twente), UIC, Railsafe
Dissemination level: **CO**

Approvals

	Name	Company	Date	Visa
WP leader	Khalid AGROU	UIC	23-09-2010	validated
WS Leader	W. v. Spronsen	ProRail	22-10-2010	validated
Project Manager	Emmanuel Buysene	UIC	26-10-2010	validated
Steering Board				

Document history

Revision	Date	Modification	Author
1.0	29-04-2009		Osmar SANTOS / York
2.0	25-06-2010	Added sections 3.2 and 3.3	Helle Hvid Hansen / Eindhoven

TABLE OF CONTENTS

List of abbreviations

ERTMS	European Railway Traffic Management System

Section 1 – EXECUTIVE SUMMARY

According to the INESS description of work, the purpose of this deliverable D.D.4.4 is to present safety requirements in a formal format. This can be done by expressing them in a mathematical language making these safety requirements precise and unambiguous. In order to achieve such a goal, two prerequisites are needed:

- Prior to formalising the safety requirements, it is necessary to have a workable description of the safety requirements. "Workable" means that they must be provided in the form of a list of statements involving the various elements of the railway domain[2] that are controlled (or not) by the Interlocking system and that are called upon to play a role in these safety requirements. Only a domain expert can derive them at this level of relevance, which is needed to formalise them.
- In order to formalise the obtained safety requirements, a formal model of the UML functional requirements of the complete[3] system itself (which we call the Interlocking Model) is also needed; otherwise, it is not possible to define the relation between the safety requirements and the model of the system.

We understand that both of these prerequisites have not been dealt with in the project so far, because:

- We have not been provided with safety requirements about the Interlocking Model at the above-mentioned level of abstraction.
- Producing a formal model of the Interlocking Model is one of the main targets of work package D.4. Indeed, this is only expected to be completed later in the project.

Due to the above, the description of work has been amended, and in this deliverable we focus on showing a methodology that could be used to obtain and formalise safety requirements. Firstly, we give a survey of academic work on the formalisation of safety requirements, and we present an example list of safety requirements that we have collected based on the surveyed papers and on communication with the railways. Secondly, we present extracts from the work carried out for the Mini Interlocking system that aimed at deriving workable safety requirements. Finally, an example of the formalisation of safety requirements using UML-B is exemplified using a small set of safety requirements derived for the Mini Interlocking system.

2 Either physical (on the field) or higher level concepts (path).

3 In the sense of an ERTMS compliant interlocking system.

Section 2 – INTRODUCTION AND APPROACH

In our current interpretation, the purpose of deliverable D.D.4.4 is to identify safety requirements in a formal format. In our current interpretation, the purpose of this deliverable is to identify safety requirements for verification, and make these precise. This document researches this topic. It collates diverse safety requirements based on existing available material suitable for formalisation, produces a methodology for future work, and gives some real examples of how they will be expressed.

2.1 Identifying Safety Requirements

The deliverable D.D.4.4 belongs to task D.4.4 which according to the INESS Description of Work consists of expressing formally all the safety requirements from the CENELEC Phase 3 documents such that they can be tested against the functional requirements modelled in UML (deliverable D.D.3.3, UIC), henceforth called the Interlocking Model. The requirements in the CENELEC Phase 3 Hazard List, however, are formulated at a level of generality that does not contain enough detail for a formalisation. A typical formulation in the Hazard List document states that an element should not work "incorrectly", without specifying what "incorrectly" (or "correctly") means.

In order to obtain a collection of safety requirements for verification we have conducted an initial study of relevant documents from the Euro-Interlocking project and the academic literature. Extracts from and references to these documents are found in Sections 3 and 4. An initial list of safety requirements was collected partly from the surveyed papers and partly through communication with ProRail. This initial list was subsequently sent to the railways for review and validation. In Section 3.1 we present the list resulting from processing the comments we received from the railways. This validation process did not determine whether the collected safety requirements cover the high-level CENELEC Phase 3 requirements.

2.2 Formalising Safety Requirements

2.2.1 Safety as Invariant

We now briefly describe how safety requirements for verification are formalised. A safety requirement can be described in terms of situations the Interlocking Model should prevent from happening. In formal verification, safety requirements are typically expressed as invariant state properties. A state property is a predicate over states in the formal model; i.e., it is a formal expression that is true of some states and false of others. A state property formally describes a situation by referring to the value of state variables. For example, if the formal model contains objects *train(1)*, …, *train(n)* representing trains, and objects *track(1)*, …, *track(m)* representing track segments, then a train object can have a property called *occupies* which takes its value from *track(1)*, …, *track(n)*. A state property formalising that no two trains occupy the same track segment can then be expressed as: $\Lambda_{1 \le i < j \le n}$ *train(i).occupies* ≠ *train(j).occupies*.

An invariant state property holds at a state *q* in a formal model if the property holds in all states reachable from *q*, where a state *q′* is reachable from *q* if in the formal model there exists an execution path from *q* to *q′*. For example, if a state property *S* formally expresses that a track segment is occupied by at most one train, then the invariant "always *S*" holds at the initial state in the formal model if *S* is true

of every state that can be reached during an execution of the formal model. An invariant can also be specified by stating what should not happen. For example, if the state property *H* formally expresses the hazard that two or more trains occupy the same track segment, then the invariant "always not *H*" holds in the formal model if in all reachable states, *H* is false.

2.2.2 Formalisation

We will use UML-B as our common language for expressing safety requirements. UML-B is one of the specification languages, which will be used for verification, cf. deliverable D.D.4.1 (Validation and Verification Strategy). UML-B has the advantage that the safety requirements and the functional behaviour can be specified in independent vocabularies, and it therefore provides a flexible yet formal language in which we can formulate and disambiguate safety requirements. Moreover, specifications in UML-B can be visualised using the graphical tools of the Rodin platform. We expect this graphical interface to be useful when communicating and discussing the requirements with our non-academic partners. At the present moment, we cannot formulate safety requirements in the property languages of the model checking tools FDR2 and mCRL2, since here a safety requirement is a formula in a logical language whose vocabulary is determined by the names of state variables and transition labels used in the formal model. By using UML-B as our common specification language, we can formulate precise safety requirements at a more general level independently of the formal model. The UML-B formalisation is easily adapted to the property languages of the tools FDR2 and mCRL2 once the formal model is available.

We point out that the creation of software tools for building the formal model is a major subtask of our verification activities (cf. D.D.4.1 Documented Strategy), and it has not yet been completed at this early stage of the project. In addition, the Interlocking Model itself (deliverable D.D.3.3) is only due in Month 28, and hence a formal Interlocking Model is only available after D.D.3.3. So even if detailed safety requirements were available to us already, they can presently not be put into the formal formats required by all the verification tools.

To demonstrate our approach, we present in this document an example formalisation in UML-B of safety requirements obtained from the document methodology "Mini Interlocking -- Hazards and Requirements" by M. Schacher (Schacher 2009) – available on the UIC extranet. M. Schacher's document presents a structured hazard analysis and derives concrete safety requirements.

2.3 Overview

The rest of the document is structured as follows. In Section 3 we first give a brief overview of the academic papers that we have used as our main sources to identify relevant safety requirements for verification, then we give a brief description of the signalling domain, and finally we present the list of safety requirements resulting from the literature survey and the communication with the railways. Section 4 contains an extract from "Mini Interlocking – Hazards and Requirements" by M. Schacher (Schacher 2009). For each safety requirement, its relevance to the verification of the Interlocking Model is indicated. In Section 5 we present an example of how the safety requirements from M. Schacher's methodology document can be formalised in UML-B. In Section 6 we summarise and give concluding remarks.

Section 3 – SAFETY REQUIREMENTS FOR RAILWAY SYSTEMS

This section presents a first set of safety requirements for verification. This set is based on a literature survey, and information obtained via discussions between the railways and the universities. The current set is intended to be a starting point for formulating specific properties in formal language that can be verified against a formal model of the common core functional requirements (INESS deliverable D.D.3.3). However, the set of requirements in this section is not claimed to be complete in any sense. Once the common core functional requirements are established we hope to be able to formulate safety requirements at a level of granularity appropriate for verification against the formal model thereof. Until this time we base our work on the functional requirements in the Generis model produced by the Euro-Interlocking project.

In Section 3.1 we give an overview of the papers surveyed. Section 3.2 contains a brief description of the signaling domain. The safety requirements are listed in Section 3.3.

3.1 Literature survey

- ***Paper 1:*** Jakob Lyng Petersen. Automatic Verification of Railway Interlocking Systems: A Case Study (Petersen 1998).

Author's Abstract:

This paper presents experiences in applying formal verification to a large industrial piece of software. The area of application is railway interlocking systems, which has earlier been addressed in, for instance, (Groote, Koom et al. 1994; Hansen 1996; Hansen 1996). We try to prove requirements of the program controlling the Swedish railway station Alingas by using the decision procedure which is based on the patented Stalmarck algorithm. While some requirements are easily proved, others are virtually impossible to manage due to a very large potential state space, which is in excess of 10^10,000. We present what has been done in order to get, at least, an idea of whether or not such difficult requirements are fulfilled or not, and we express thoughts on what is needed in order to be able to successfully verify large real-life systems.

Relevance:

A general paper on railway formal methods using an approach called STERNOL and theorem proving. Not particularly strong on safety requirements, but nevertheless it is of interest.

- ***Paper 2:*** Dines Bjørner. Formal Software Techniques in Railway Systems (Björner 2000).

Author's Abstract:

We discuss a role for the use of formal techniques in understanding the domain of railways, in expressing requirements to, and in the design of trustworthy software for the backbone and program packages for railways. We exemplify such techniques as applied to the recording of our domain understanding and hints at their use in requirements and design. We motivate the use of formal techniques, and survey, ever so briefly, the state-of-affairs of such use. We finally put forward a proposal for and invite you to join such a proposed European Railway Software Technology Research & Development project: TRaIn: The Railway Infrastructure.

Relevance:

This paper provides a framework for modelling railway entities and layouts. A simple framework and topographical model such as this could form the basis for specifying safety properties.

- ***Paper 3:*** Kirsten M. Hansen, Anders P. Ravn, and Victoria Stavridou. From Safety Analysis to Software Requirements (Hansen, Ravn et al. 1998).

Authors' Abstract:

Software for safety critical systems must deal with the hazards identified by safety analysis. This paper investigates how the results of one safety analysis technique, fault trees, are interpreted as software safety requirements to be used in the program design process. We propose that fault tree analysis and program development use the same system model. This model is formalised in a real-time, interval logic, based on a conventional dynamic systems model with state evolving over time. Fault trees are interpreted as temporal formulas and it is shown how such formulas can be used for deriving safety requirements for software components.

Relevance:

Shows a rigorous process for deriving safety requirements using safety-analysis techniques (fault trees in this case). It uses a railway layout and interlocking as a case study. The final expression of requirements should aim to demonstrate a link to the top-down safety analysis. This is also covered in part by the Mee-Ingleby paper (Paper 6, below).

- ***Paper 4:*** Neil Robinson and George Nikandros. Railway Signalling Design Tools - Supporting Control Table Designers (Robinson and Nikandros 2003).

Authors' Abstract:

Control Tables are the functional specification for railway signalling interlockings. They have an important role in the signalling design process since they act as an agreement between the railway administration and the train operators on when moves will be permitted on a track layout. They also act as a design specification for use by the interlocking designers and a test specification for use by testers. Control Tables contain the key functional safety requirements for the interlocking. We report on research into a new Signalling Design Toolset for supporting the production of Control Tables. The main functions of the new toolset are to automatically generate Control Tables, to support the editing of Control Tables, and to automatically verify Control Tables. Previous work on supporting signalling design has mostly been aimed at supporting the production of configuration data for electronic interlockings. By aiming at improvements at the Control Table design stage, we intend to eliminate errors as early as possible in the signalling design process, and thus reduce rework.

Relevance:

The paper contains a section on an approach to formalising safety requirements. The paper addresses the key area of application data in a real scheme. It also discusses approaches to model checking and formalising requirements that may be of benefit to other V&V research.

- ***Paper 5:*** Lars-Henrik Eriksson. Specifying Railway Interlocking Requirements for Practical Use (Eriksson 1996).

Authors Abstract:

An essentially complete formal specification of safety requirements for railway interlocking has been developed. The work is part of a project with the Swedish National Rail Administration investigating the feasibility of using formal methods for the analysis of interlockings in a production setting. An overview of the specification is given and two ongoing case studies on verifying interlockings using the specification are described. Verification is done using the very fast Stalmarck theorem prover for propositional logic. The current limits of the technology are discussed.

Relevance:

The paper is not very strong on the theoretical underpinning of safety requirements, but it is included since it represents a key industrial case study. Furthermore, it references other work done by Swedish railway authorities, and so this may be accessible via partners if needed.

- ***Paper 6:*** M. Ingleby and D.J. Mee. A calculus of hazard for railway signalling (Ingleby and Mee 1995).

Authors' Abstract:

Interlockings - systems which control railway signals - are modelled as situated automata holding in memory an image of their trackside environment. Interlocking functionality is generic, but each interlocking consults a geographic database which specifies the topography of its environment. A calculus of hazard comprising a theory of trackside geography and generic state of trackside environments is set up using a predicate calculus based on incidence relations. The calculus is sufficiently expressive for the articulation of hazard defence rules - which are obtained from a typical IEC fault modes and effects analysis (FMEA) - free of area-specific reference. Safety of an interlocking is formulated as an NP-complete proof problem expressing the invariance of a set of hazard defence predicates of the calculus. A scalable approach to this proof problem is developed by representing a signalling area as a set of weakly interacting localities of low combinatorial complexity. The approach uses Galois connection tools borrowed from formal concept analysis.

Relevance:

This is perhaps the most relevant work, stemming from an industrial and academic collaboration in the 1990's sponsored by British Rail Research. It provides a number of key safety requirements in a formal notation that can be directly correlated with the INESS domain. The hierarchy of these safety requirements is depicted in a block diagram based on accident prevention.

- ***Paper 7:*** Andrew Simpson, Model Checking for Interlocking Safety (Simpson 1998).

Author's Abstract:

We describe how the technique of model checking has been applied to the verification of geographic databases which are associated with Solid State Interlocking railway signalling systems. We represent such databases in terms of Communicating Sequential Processes (CSP) and verify their safety using the refinement checker FDR2. The approach to compositionality and refinement which is employed by CSP allows us to reduce the analysis of safety invariants for geographic databases to a number of small mechanical tests.

Relevance:

Simpson lists five key safety invariants (Sect. 2.3) that are synonymous with key UK safety requirements for an SSI interlocking. Despite being technology specific, the safety requirements correlate broadly with more generic interlocking function. Furthermore, Simpson discusses approaches to model checking that may be of interest to the V&V research. There is also a related doctoral thesis.

- ***Paper 8:*** Anne Haxthausen and Jan Peleska. Formal Development and Verification of a Distributed Railway Control System (Haxthausen and Peleska 2000).

Authors' Abstract:

In this article, we introduce the concept for a distributed railway control system and present the specification and verification of the main algorithm used for safe distributed control. Our design and verification approach is based on the RAISE method, starting with highly abstract algebraic specifications which are transformed into directly implementable distributed control processes by applying a series of refinement and verification steps. Concrete safety requirements are derived from an abstract version that can be easily validated with respect to soundness and completeness. Complexity is further reduced by separating the system model into a domain model and a controller model. The domain model describes the physical system in absence of control and the controller model introduces the safety-related control mechanisms as a separate entity monitoring observables of the physical system to decide whether it is safe for a train to move or for a point to be switched.

Relevance:

This paper covers distributed control and shows how simple safety requirements can be derived. It uses the RAISE method, an alternative notation to more common formal methods. This may be useful in terms of the GENERIS model, which has a hardware abstraction layer and this needs to be considered in terms of safety properties and timings.

- ***Paper 9:*** M.J. Morley. Safety in Railway Signalling Data: A Behavioural Analysis (Morley 1993).

Author's Abstract:

Higher Order Logic is used to analyse safety properties of a computer based railway signal control system. British Rail's Solid State Interlocking is a data driven controller whose behaviour is governed by rules stored in a geographic database. The correctness of these data are highly critical to the system's safe operation. Taking as our starting point Gordon's implementation of program logics in Higher Order Logic, we formalise the data correctness problem in a natural way, designing tactics to automate much of the verification task. Our approach requires quadratic time and linear space, and is thought to be adequate for the current generation of Solid State Interlockings. Properly formalised, the compositional proof strategy we suggest will allow us to scale up our analyses arbitrarily.

Relevance:

Similar in scope to the Simpson paper (Paper 7, above), it covers SSI data constructs. It describes a number of key safety properties that are proved using Higher Order Logic. Morley has also produced a doctoral thesis on the same subject.

- ***Paper 10:*** Tomas Hlavaty, Libor Preucil, Petr Stepan. Case Study: Formal Specification and Verification of Railway Interlocking System (Hlavaty, Preucil et al. 2001).

Authors' Abstract:

The contribution addresses the problem of software life-cycle, aspects and application of formal methods in functional specification, design and verification of real-time software systems in safety-critical applications. The target application there under aims to verify the critical parts of the designed interlocking system. The higher safety, reliability and minimised costs of the design can be achieved using presented techniques. Bidding for this a synchronous data-flow language Lustre has been used to verify safety properties of selected parts of the system. The introduced test-case has been implemented on distributed architecture of multiple single-chip microprocessors.

Relevance:

This is less relevant for eliciting safety requirements, but provides a strong focus on the control system and real-time data acquisition.

- ***Paper 11:*** W.J. Fokkink. Safety criteria for the vital processor interlocking at Hoorn-Kersenboogerd (Fokkink 1996).

Authors' Abstract:

We formulate several classes of safety criteria for railway yards in terms of observable behaviour. These criteria are meant to protect trains from collisions and from derailments. We identify a number of safety criteria, and present instances of these classes for the case of the railway yard at station Hoorn-Kersenboogerd. These criteria have all been checked by means of the Stalmarck theorem prover, using a methodology from Groote, Koorn, and Van Vlijmen.

Relevance: This paper is highly relevant as it gives several safety requirements, both informal generic ones, and instantiations of these in formal notation for a specific railway yard.

- ***Paper 12:*** A. Cimatti, F. Giunchiglia, G. Mongardi, D. Romano, F. Torielli, P. Traverso. Formal Verification of a Railway Interlocking System using Model Checking (Cimatti, Giunchiglia et al. 1998).

Authors' Abstract:

In this paper we describe an industrial application of formal methods. We have used model checking techniques to model and formally verify a rather complex software, i.e., part of the safety logic of a railway interlocking system. The formal model is structured to retain the reusability and scalability properties of the system being modelled. Part of it is defined once for all at a low cost, and re-used. The rest of the model can be mechanically generated from the designers' current specification language. The model checker is hidden from the user, it runs as a powerful debugger. Its performance is impressive: exhaustive analysis of quite complex configurations with respect to rather complex properties are run in the order of minutes. The main reason for this achievement is essentially a carefully designed model, which exploits all the behaviour evolution constraints. The reusability/scalability of the model and the fact that formal verification is automatic and efficient are the key factors which open up the possibility of a real usage by designers at design time. We have thus assessed the possibility of introducing the novel technique in the development cycle with an advantageous costs/benefits relation.

Relevance:

The focus of this paper is on the verification and the integration of formal methods into the software development process. Still, the paper provides four concrete safety requirements in Section 6 that seem to complement other safety requirements found in the literature.

We conclude this section by listing a wide variety of papers involving the formalisation of railway safety requirements: (Hall 1990; King 1994; King 1994; Morley 1996; Raili 1996; Simpson, Woodcock et al. 1997; Cimatti, Pieraccini et al. 1999; Bozga, Fernandez et al. 2000; Gnesi, Latella et al. 2000; Blanco 2001; Huber 2001; Winter 2002; Cavalcanti, Sampaio et al. 2003; Winter and Robinson 2003; Faulkner 2004; 2006; Abrial 2006; Freitas, Woodcock et al. 2006; Winter, Johnston et al. 2006).

3.2 Signaling domain description and terminology

This section provides a brief description of a few key features of railway signaling and interlocking functionality. These descriptions are based on the definitions in the documents *Unified Glossary of Terms Report* (INESS deliverable D.D.1.1), *Catalogue of Commands and Statuses* (INESS deliverable D.D.1.4) and on the Generis model.

This section is mainly included for the benefit of non-railway specialists such as the universities. It is not intended to be a complete or comprehensive description, but only to provide a context for understanding the safety requirements listed in Section 3.3.

Signals:

Signal types	*in Generis:*
main (or running)	yes
line block signal	yes
shunting	yes
warning (or distant)	no
repeater	no

Signal aspects in Generis: stop**,** DOS proceed**,** proceed**,** canceled.

Routes:
A *route* is a predetermined path for track movement. The route body is the part of the route between the entry signal and exit signal, and does not include the flank or overlap sections. Note that a route does not need to go through a point.

Conflicting routes: A route R1 is in conflict with another route R2, if R1 crosses, converges with or opposes R2, or would require the use of a route element which is used by R2 in a position which is incompatible or conflicting with the position required for R1. See Figure 3.1 for example illustrations.
(a) R1 and R2 cross each other.
(b) R1 and R2 converge.
(c) R1 and R2 do not overlap, but are still in conflict.
(d) R1 and R2 are opposing (entry of R2 is the first opposing signal of the entry of R1).
(e) R1 and R2 are opposing (entry of R2 is the second opposing signal of the entry of R1).

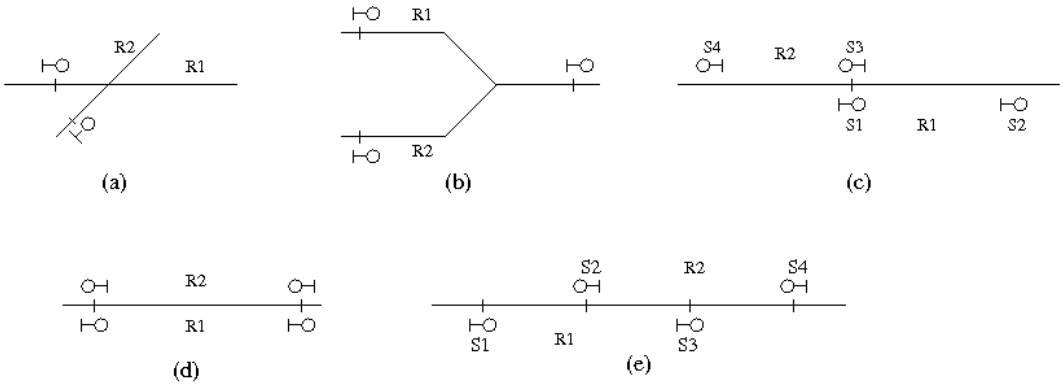


Figure 3.1

In The Netherlands, Figures 3.1(d) and 3.1(e) are considered two different types of conflict. In Figure 3.1(e), suppose you want to set the route going from S1 to S3. Then it must be checked that no route has been set from the first opposite signal S2, and also that no route has been set from the second opposite signal S4 to S2.

Route setting: The typical phases in route setting are:

1. route request
2. availability check
3. reserve elements (points etc.)
4. position points
5. receive confirmation from points
6. lock route incl. lock points
7. set control signal to proceed
8. apply approach locking
9. monitor train in section
10. release route elements

Note: A “route is set” means that the route is reserved or locked.

Approach locking: Approach locking of signals is a special form of locking which is only used in connection with cancellation or changing of routes. When a route is cancelled, its entry signal must immediately show stop, however, if in the approach of the route, some track section is occupied, then the route must stay locked for a certain amount of time T. In the United Kingdom and the Netherlands, T is normally 2 minutes (cf. Paper 6 of the previous section). In Spain T is equal to the time release for the movement authority of ETCS). The purpose of approach locking is to prevent an approaching train from passing over a non-locked route in case the train is too close to stop at the entry to that route. Hence, an approach locked route is locked, but its entry signal maintains a stop aspect for 2 minutes. During these 2 minutes, the entry signal is said to be approach locked.
In some countries (e.g. the United Kingdom) the approach locking timer may be replaced with more complex interlocking functionality that actually monitors the position of a train (called “comprehensive approach locking”). This feature improves operational flexibility so that routes can be released earlier if the interlocking establishes the train has come to a safe stand (sometimes this feature is necessary in complex station areas where a two minute wait may be restrictive).

Flank protection: An element implements flank protection for a route if the element is positioned and locked to prevent collisions on overrun or converging routes in case a train ignores a red light. In Generis, flank protection is implemented by signals, points and derailers. For example, in Figure 3.2 below, suppose the route R1 from signal S1 to signal S2 is set. Point P1 implements flank protection for R1 by being positioned right and locked, such that if a train should pass a red light at S3, the train is

guaranteed not to collide with a train in R1. Moreover, the points P1 and P2 should be reserved, positioned, locked and released at the same time.

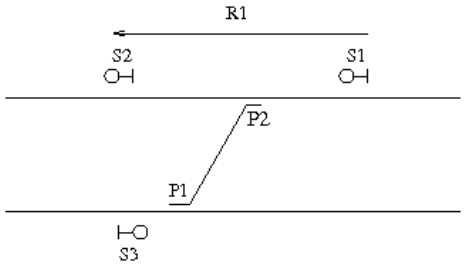


Figure 3.2

There are some UK particularities whereby some 'flank' points must be set and detected (in the safe position), but not locked otherwise other routes could not be set and this would be operationally restrictive.

Level crossings:
The activation zone of a level crossing X with respect to a route R consists of those track sections on R from which a train going at maximum speed can reach X within the approach time. When a route R is set over a level crossing, the level crossing process carries out a search for occupied tracks in the activation zone of X with respect to R.

Timing constraints:
Most requirements allow the system a certain amount of time to reach a “safe” state. On the other hand, the system must also react within certain time limits when unsafe situations arise, and certain functionalities, such as approach locking and activation of level crossing barriers, depend crucially on time parameters. An analysis of key timing-related constraints is, however, beyond the scope of this document. For that reason, (non-quantitative) timing constraints are included only in some formulations of safety requirements in Section 3.3.

3.3 List of safety requirements

1. Aspect sequences: A stop aspect must be preceded by a warning aspect to provide drivers sufficient time to brake:
    a. If S1 is the signal immediately in rear of S2 and S2 shows stop, then S1 shows stop or a warning aspect.
2. Routes:
    a. If the entry signal of a route does not show stop, then the route is locked.
    b. If two main routes are in conflict, then the entry signal of at most one of them shows proceed.
    c. The condition for locking a route is that its elements are detected in the correct positions and locked. If a route is locked and one of its elements changes from correct to incorrect position the route should stay locked.
3. Line block signals and track occupation:
    a. If a line block signal shows proceed, then all track sections in advance of and affected by the signal in the block are unoccupied, and all blocking conditions are matched.

4. Points: Let P be an arbitrary set of points and let S be the signal in the rear of P.
    a. P is only allowed to move if P is unoccupied and unlocked.

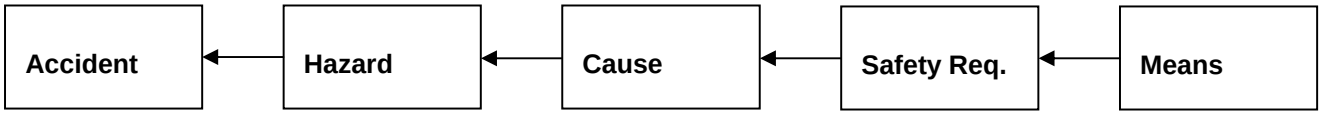
   b. If P is not locked, then no route over P is set.
   c. If P is not locked, then S shows stop.
   d. If P is not locked, then S is not approach locked.
   e. If more than T time units have passed since P last confirmed its position, then S shows stop.
   f. If P is not locked and its associated TVP is occupied, then P can be moved by an emergency command (ADIF).
5. Level Crossings:
   a. If the track section at a level crossing is occupied, then the level crossing is closed.
   b. If a track section is occupied inside an activation zone of the level crossing, then the level crossing barriers must be closed within T time units.
6. Approach locking:
   a. If a route was cancelled less than a T time units ago, then the route is locked.
7. Flank protection:
   a. If a point P implements flank protection for a route R and R is locked, then P is positioned (and locked).
   b. If a signal implements flank protection for a route R and R is locked, then S shows stop (and is locked).
   c. If a derailer D implements flank protection for a route R and R is locked, then D is positioned on-rail (and locked).
8. Partial route release:
   a. If a train passed the entry signal of a route R less than T time units ago, then a partial route release of R command cannot be carried out.

Section 4 – SAFETY REQUIREMENTS OBTAINED FROM THE EURO-INTERLOCKING PROJECT

In addition to the safety requirements obtained from the literature study described in Section 3, we have found the document “Mini Interlocking – Hazards and Requirements” (Schacher 2009) to be of particular relevance to the safety requirements identification. Its relevance to this document is two-fold. On the one hand, it illustrates the complexity of the requirements analysis and the amount of domain knowledge necessary to carry out this analysis. On the other hand, it provides us with an initial set of informal safety requirements that could be used for the Interlocking Model. We also use these requirements to exemplify our approach to formalisation in Section 5.

In the rest of this section, we present an extract of (Schacher 2009). We have intentionally chosen to leave out the parts dealing with quantitative aspects such as severity, frequency and risk evaluation, as these are not relevant for the derivation of safety requirements. Furthermore, some safety requirements are beyond the scope of the Interlocking Model. These include speed control and the correct adherence to signals by drivers. In the descriptions of safety requirements and means identification, we have included some comments that try to relate the particular relevance of the safety requirement and means identification with respect to the Generis model. Generis is our current example interlocking model which is assumed to be a close approximation of the final (still unavailable) Interlocking Model.

4.1 Hazard Analysis



The above diagram illustrates the steps in the causal analysis of safety requirements and the means intended to ensure the requirements hold. The analysis starts from (generally formulated) Accidents, derives more concrete causal effects, and ends with the behavioural features (Means) that should be part of the interlocking system’s behaviour in order to prevent accidents.

4.1.1 Accident Identification

A-1	Train <=> train collision	either facial or non-facial
A-2	Train <=> person collision	e.g., a single person or a car on a track segment
A-3	Train <=> object collision	e.g., rocks on a track segment
A-4	Train derailment	

4.1.2 Hazard Identification

H-1	Two or more trains are on the same track segments.	A-1		Potentially legitimate.
H-2	A point is moving under a train.	A-4		Must not happen.
H-3	Obstacle on a track segment.	A-2, A-3, A-4		Only trains can be detected on track segments.
H-4	Train speed too high.	A-4		Cannot be detected.
H-5	Mechanical defect of track element.	A-4		Will be reported.
H-6	A point leads a train to an incorrect track segment.	A-1, A-2, A-3		Must not happen within a route.

4.1.3 Causal Analysis

C-1	**Interlocking system**	**---**	**---**	
C-1.1	Wrong steering command.	H-1, H-2	Software	
C-1.2	Late steering command.	H-1, H-2	Software	
C-1.3	Missing steering command.	H-1, H-2	Software	
C-2	**Track segment**	**---**	**---**	
C-2.1	Sensor defect.	H-1	Hardware	
C-2.2	Occupied by undetectable obstacle.	H-3	System	e.g., a person or some rocks
C-3	**Point**	**---**	**---**	
C-3.1	Point in wrong position.	H-1	C-1.*, C-3.2, C-3.3	
C-3.2	Sensor defect.	C-3.1	Hardware	
C-3.3	Actuator defect.	C-3.1	Hardware	e.g., motor defect
C-3.4	Manual operation of point.	H-1, H-2	System/Human	
C-4	**Signal**	**---**	**---**	
C-4.1	Wrong signal aspect.	H-1	C-1.*	
C-4.2	Missing signal aspect.	H-1	Hardware	e.g., lamp defect
C-5	**Reserved paths**	**---**	**---**	
C-5.1	Track elements belong to more than one reserved path.	H-1	C-1.*	
C-6	**Train**	**---**	**---**	
C-6.1	Train too fast for point.	H-1	System	
C-6.2	Train too fast for signal.	H-1	System	
C-6.3	Train ignores signal.	H-1	Human	
C-7	**Environment**	**---**	**---**	
C-7.1	No fence around tracks	H-3	C-2.2	

4.1.4 Safety Requirements

S-1	Not more than one train shall occupy a track segment.	H-1	Unless otherwise configured.
*S-2	A point shall not move if its associated track section is detected as being occupied.	H-2	Is in the Generis model.
*S-3	A train shall not be led to a track segment with an obstacle.	H-3	The only detectable obstacle is another train and it may be legitimately present there.
S-4	A train shall not be too fast for a track element.	H-4, C-6.1, C-6.2	Not relevant in the context of the current interlocking specification, as train speed is not measured.
*S-5	A train shall not be led to a track element known to have a mechanical defect.	H-5	No routes over failed elements (unless configured otherwise).
*S-6	A train shall not be led to an occupied track element.	H-5	Not true, it is permitted (even required) in certain DOS routes.
*S-6	No point shall be in an incorrect position for a reserved path when a train is in the route of that reserved part.	H-6	That is not really the problem. No signal shall be set to proceed for a given route which has points not detected and locked in the correct position.
*S-7	A state of a track element shall not be assumed before it is acknowledged.	C-1.*, C-3.*	The interlocking's 'perception' of the status of external track elements is always the result of a detected value being received.
S-8	Track element defects shall no be left undetected.	C-2.1, C-3.2, C-4.2	All elements shall have a defined failure state.
S-9	Obstacles shall not be left undetected.	C-2.2	Trains can be detected, but not much else.
*S-10	Any track segment shall not belong to more than one reserved path whose entry signal is showing proceed.	C-5.1	Depends on configuration (flank can be shared).
S-11	Ignorance of a signal by a train shall not lead the train into another reserved path.	C-6.3	Passing the signal cannot be prevented.
S-12	A simple failure shall not lead to an accident.	C-6.3	Simple is not defined above. However, this is just the kind of thing we wish to discover via verification.
*S-13	A signal shall not be switched to stop, if a train is approaching it within a certain distance.	C-6.2	Not true. It can be put to stop, it is the route that cannot be released immediately if the train is within a certain distance.

[4] Requirements with an asterisk in their Id may be formally validated in the UML model

*S-14	An entry signal shall not show proceed, if one of the track segment(s) within a certain distance after the entry signal at the end of the reserved path are occupied.	C-6.2, C-6.3	An entry-signal will never show proceed if any element in the route body is occupied unless the route is configured as a DOS route.

4.1.5 Means Identification

M-1	Track segments shall be associated to reserved paths.	S-1	Only if so configured.
M-2	A reserved path shall be protected by an entry signal.	S-1, S-5, S-6, S-7	True.
M-3	Points in a reserved path shall be locked against accidental movements.	S-2	Point in route bodies are always locked and in overlaps are generally locked.
M-4	Obstacles on track elements shall be detected.	S-3	Only if it leads to “<dv> detected occupied” (which it will not).
M-5	The allowed maximum speed shall be indicated to the train driver.	S-4	Not by this Interlocking specification.
M-6	Mechanical defects on track elements shall be detected.	S-5, S-6, S-9, S-10	This is in the Generis model.
M-7	Detected values from track elements shall be used to reflect their current state.	S-8	This is in the Generis model.
M-8	A command's acknowledge shall be monitored by a timeout.	S-9	For points, the *execution* of the command is monitored by a time out.
M-9	A track element shall be reserved for at most one reserved path whose entry signal is showing proceed.	S-11	Depends on the type of route, flank can be shared between routes.
M-10	A train passing a signal showing stop shall be detected and stopped.	S-12	Track occupation by a train can always be detected, thus ‘Illogical’ occupation of a set route with a proceed aspect can be too.
M-11	All steering of points (incl. manual steering) shall be done via the interlocking.	S-2	This is in the Generis model.
M-12	Double lamps shall represent signal aspects.	S-13	Irrelevant to the Generis model.
M-13	An approach area shall be placed in front of a signal.	S-4, S-14	If so configured in the route.
M-14	An overlap area shall be placed after a signal.	S-4, S-15	If so configured in the route.
M-15	A reserved path shall be flank protected.	S-1	If so configured in the route.
M-16	Fences shall protect high-speed track segments.	C-7.1	Irrelevant.

Section 5 – FORMALISATION OF SAFETY REQUIREMENTS USING UML-B

In this section we introduce an example formal model to formalise the top level safety requirements for Train Interlocking Systems. The formalisation process starts from the Hazard analysis described in the previous section for the Mini Interlocking System (Schacher 2009).

Accident Identification

Id	Description	Severity[1]	Comments
A-1	Train <=> train collision	4	either facial or non-facial
A-2	Train <=> person collision	3	e.g. a single person or a car on a track segment
A-3	Train <=> object collision	2	e.g. rocks on a track segment
A-4	Train derailment	2-3	

Hazard Identification

Id	Description	May Cause	Frequency	Comments
H-1	Two or more trains are on the same track segments.	A-1		
H-2	A point is moving under a train.	A-4		
H-3	Obstacle on a track segment.	A-2, A-3, A-4		Only trains can be detected on track segments
H-4	Train speed too high.	A-4		
H-5	Mechanical defect of track element.	A-4		
H-6	A point leads a train to an incorrect track segment.	A-1, A-2, A-3		

The model uses the UML-B notation which is a graphical front-end for the Event-B language and tools. The Rodin tool set includes a prover, which automatically attempts to prove properties about a model whenever it is saved. The properties that are proved include invariant preservation (which may include the safety invariants we are interested in here as well as simple typing properties) and that a refinement still behaves in a way that was permitted by the model it refines. Proofs may not succeed, either because the model is incorrect or because the proof is too difficult for the automatic prover. The Rodin toolset includes an interactive prover, where the user can attempt to guide the automatic prover. The model was automatically proven using the Rodin tool set[5]. The model consists of the following three refinement steps:

- Introduce domain concepts required to describe the 6 hazards above. The idea of this first stage is to describe what would happen if the system was not made to behave safely. If we animated this model we would observe trains crashing and being derailed. The point of this stage is just to introduce the minimal 'vocabulary' that will be used in later models.
- Introduce assumptions relating to domain concepts not controlled by the interlocking systems and limitations of the interlocking system. The point of this stage is to explicitly describe the safety properties that won't be dealt with by the interlocking system. This can include assumptions we make about things outside the control of the interlocking system, for example, the actions of a driver or hazards that the interlocking system cannot address and therefore have to be risked. By modelling these things explicitly we are forced to consider them and allow domain experts the chance to object to them.
- Introduce safety requirements of the interlocking system. This final stage addresses the safety

[5] One proof obligation of the feasibility of an initialisation required a small amount of interactive assistance.

requirements that the interlocking system must satisfy. They are expressed as invariant properties that must always hold and a behavioural model that satisfies these invariants. This behavioural model could be further refined until an interlocking system is represented. The Rodin proof tools will ensure that this interlocking system is a safe system.

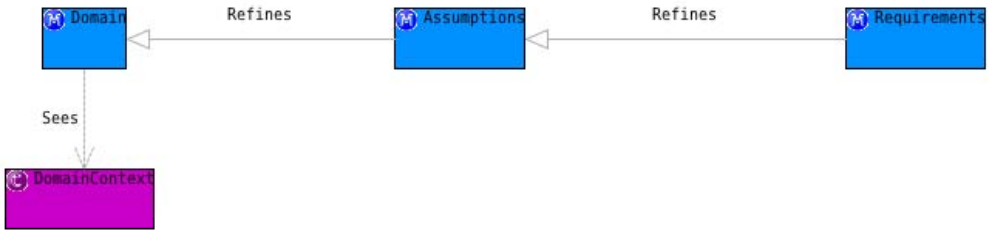


Figure 5.1 - The components in the model: three refinement steps and a context.

This model is deliberately at a high level of abstraction. It does not describe the functionality of an interlocking system. The intention is to describe just enough detail in order to express the safety hazards that must be avoided. This is done using some abstract state variables and some events that alter those variables. These events are then given guards (conditions that are necessary before the event can occur). In this way, constraints on the behaviour are modelled in a way that is expected to satisfy the safety requirements. The safety requirements are then expressed as invariants (properties of the state variables that should hold at all times). It is then proved, using the Rodin verification tools, that no sequence of events can reach a state that violates the invariants.

5.1 Domain Concepts

At the first level we model the domain in (only just) sufficient detail to be able to express the hazards. In this level we just introduce the domain concepts (state-space) and do not constrain the system to behave safely.

For H1, we need a set of trains, a set of track segments and a relationship occupies between them. Since implicitly, trains move about on the track, we allow trains to change their *occupies* value to the next track (next is introduced in H2 below).

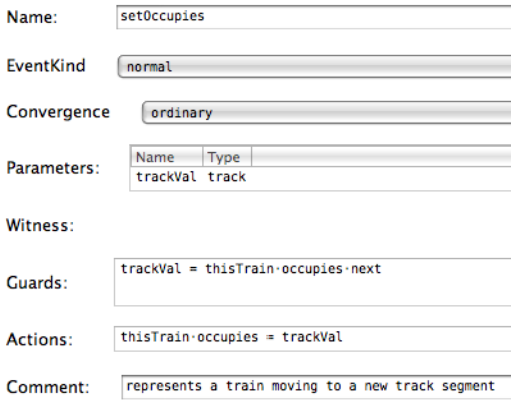


Figure 5.1 - Specification for the event 'setOccupies'. The value of the parameter trackVal is nondeterministically chosen from its type (here 'track') but this choice is further constrained by the guards to be the 'next' track from the one the train currently occupies. [n.b. EventKind is discussed in a later example. Witness and Convergence concern refinement of parameters and new events, neither of which are used in this document].

For H2, we need the concept of changing points. We model this by giving track a 'next' relationship to another track and allowing it to change. I.e. all track segments are potentially points although, in reality, most never change.

For H3, we give track a Boolean attribute, 'obstructed' and allow this to be altered nondeterministically.

For H4, we give 'train' a Natural attribute, 'speed' and allow this to change nondeterministically. Although not made explicit in the hazard, the concept 'safe speed' depends on the occupied track segment, so we also introduced a constant attribute 'safeSpeed' for the track. This is done in the associated context.

Figure 5.2 - Domain Context. In UML-B, any explicitly defined sets and constants are specified in a separate component called a context. Here, we just need to specify that there is a set of track sections called 'TRACK' and each one has a constant which represents the safe speed for that track section. For modelling convenience we also define special instance of TRACK called 'nullTrack' to represent non-existent track.

For H5, we give track a Boolean attribute 'failed' and allow this to change nondeterministically.

For H6, we give track a Boolean attribute 'incorrect' and allow this to change nondeterministically. Note that we take a slightly different interpretation of 'incorrect' than in the hazard analysis. The hazard analysis seems to treat incorrect as a general term including track that is occupied, obstructed or is in use by people or cars. The first two meanings are already covered by H1 and H3 respectively. Since people or cars using the track are either treated as obstructions (H3) or are using a track device that is controlled by the interlocking system. Therefore, we reduce incorrect to mean track sections that have an associated device that is in a state where it cannot be safely occupied. This may be an incorrect

interpretation but through discussion with domain experts this could be clarified and fixed if necessary. An important outcome of formalising the safety requirements is that it forces us to confront these ambiguities and encourages them to be clarified.

Note that, for modelling convenience, we treat track as a fixed set of instances with a configuration whereas we treat trains as a variable set of instances so that the train's attributes can be easily initialised to a valid and consistent state, which is refined in later steps.

For this domain model, 17 proof obligations were generated by the Rodin tools. Of these, sixteen were discharged automatically and one required some interactive assistance.

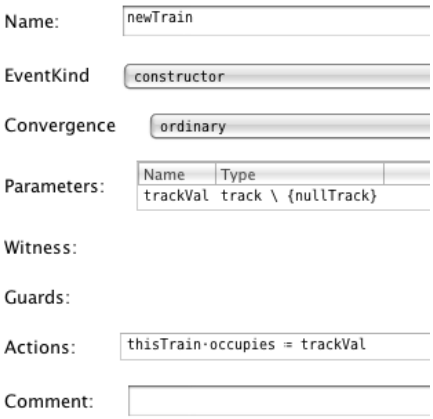


Figure 5.3 - Specification of the newTrain event which adds a train to the system. The EventKind is set to 'constructor' to represent this behaviour. A parameter trackVal picks any track segment (other than nullTrack) and this is used to initialise the train's track position.

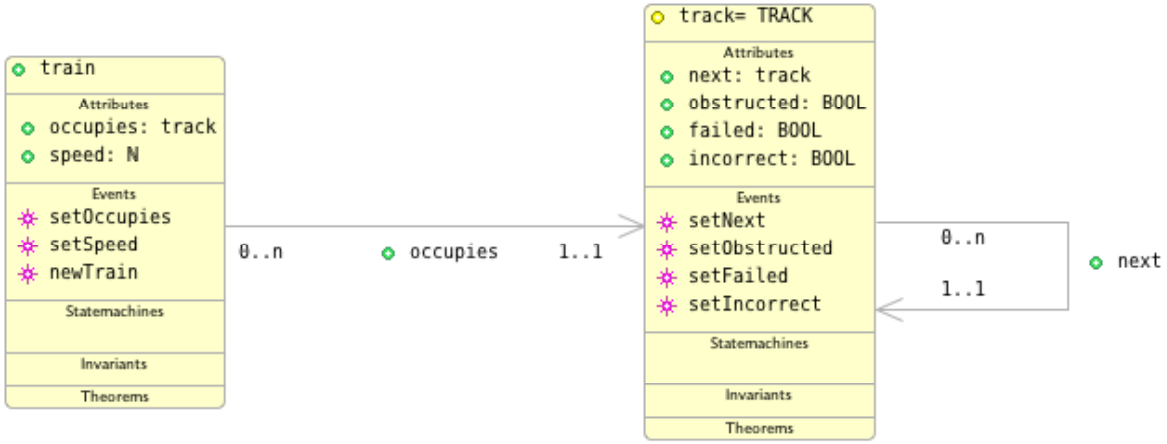


Figure 5.4 – The Domain model shows the main features needed to consider the given hazards. This consists of the two classes, train and track and their attributes, associations and events. Note that the instances of class track are linked to the set TRACK that was defined in the context.

5.2 Excluded from Scope of Interlocking System

In the next level we introduce assumptions about the things that are not covered by our interlocking system. This includes things that are outside of the scope of interlocking and cannot be influenced by it, as well as limitations of the interlocking system.

5.2.1 Assumptions

For H3, obstacles are not detected by the interlocking system. Therefore we assume that no obstacles will ever be present by introducing a guard into the 'setObstructed' event so that 'obstructed' is only ever set to FALSE. (In the previous level, boolVal was the nondeterministically selected Boolean parameter representing the new value of obstructed).

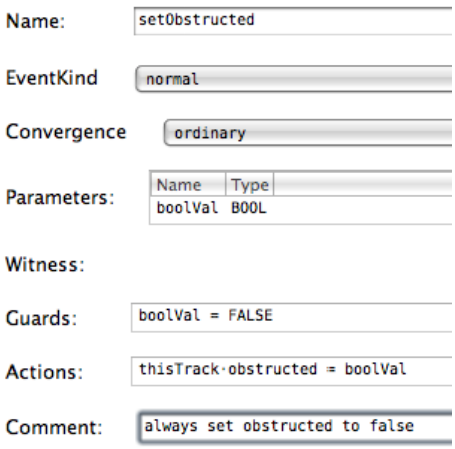


Figure 5.5 – Specification for the event setObstructed. The guards force the new boolean value of obstructed to always be false.

For H4, speed is outside of the scope of the interlocking system. There is nothing in the interlocking system that can influence the speed of the train and we therefore make the assumption that the driver only selects safe speeds. We implement this assumption by introducing a guard into the setSpeed event that restricts the new speed (natVal) to be less than the safe speed of the currently occupied track.

natVal < thisTrain occupies safeSpeed

We also assume the driver will slow down before moving the train into a new track that has a safe speed limit that is lower than the current train speed. We implement this assumption by introducing a guard into the setOccupies event that prevents the new track segment being occupied unless the train's current speed is lower than the new track segment's safe speed.

thisTrain speed < thisTrain occupies next safeSpeed

5.2.2 Limitations

For H5, although the interlocking system can prevent a train from moving to a track section that is failed, it cannot prevent failures from occurring. Therefore, it is a limitation of the system that it cannot prevent a track from failing when it is already occupied. It is assumed that this is so unlikely that it can be assumed not to occur. (If necessary, it could be made more unlikely by providing redundant systems).

To represent this assumption we introduce a guard into the setFailed event so that track sections cannot fail while they are occupied. Note that the guard only prevents occupied track from failing, it does not prevent failures from being fixed (boolVal = FALSE). [Range (ran) of a function is the set of all target instances that are currently mapped to by the function. Hence ran(occupies) is the set of track segments that are occupied by trains.]

thisTrack ran(occupies) boolVal = FALSE

5.2.3 Invariants

We can now introduce the safety conditions for these hazards as invariants on the properties of a train. (Since they are placed within the class train, they apply to all instances of trains, i.e. universal quantification is implicit for all thisTrain belonging to train).

For H3, we introduce an invariant to say that an occupied track is never obstructed.

thisTrain occupies obstructed = FALSE

For H4, we introduce an invariant to say that a train's speed is always less than the safe speed of the track it occupies.

thisTrain speed < thisTrain occupies safeSpeed

Initially the Rodin tools generated twelve proof obligations for this stage of the model. At the first attempt, the automatic prover, failed to prove two proof obligations. Examining these two proof obligations (e.g. `obstructed(next(occupies(thisTrain)))=FALSE`) we observed that they both concern the invariant associated with H3. We are fairly confident that this invariant is satisfied since we added a guard to ensure that all track segments are always unobstructed. However, the prover doesn't know this information. We could attempt to assist the prover interactively, but in this case it is easier to add an invariant to the model to give the prover the additional information it needs so that it can break the proof in to smaller steps. The prover will first prove that this new invariant is true and then use it to prove H3. After adding this invariant, 15 proof obligations were generated by the Rodin tools, but all were then discharged automatically.

thisTrack obstructed = FALSE

Figure 5.6 – The refined model consists of the two refined classes with all their previous attributes and associations inherited (i.e. retained) and some new invariants added.

By omission we assume that the remaining events are fully controlled by the interlocking system that is yet to be introduced. More explicitly, the interlocking system has full control over when trains can change their occupancy of track sections, when points can change (the value of next) and when a track section becomes 'incorrect'.

5.3 Safety Requirements

In this level we introduce the safety requirements for the interlocking system as invariants and then constrain our model with additional guards needed to satisfy them.

For H1, we introduce an invariant to say that occupies is injective. (An injective function is one where at most one element of the domain maps to each element in the range. I.e. only one train occupies a particular track element).

occupies train track

For H2, we introduce an invariant to say that the track occupied by a train has the same value for next as it did when the train first occupied the track. (Note that, since this is a temporal property, we need to introduce some extra data, nextWhenOccupies in order to express this as an invariant.)

thisTrain nextWhenOccupies = thisTrain occupies next

For H5, we introduce an invariant to say that the track occupied by a train is not failed.

thisTrain occupies failed = FALSE

For H6, we introduce an invariant to say that the track occupied by a train is not incorrect.

thisTrain occupies incorrect = FALSE

We now need to constrain the behaviour of our model so that it satisfies these invariants.

For H1, we introduce a guard to setOccupies that represents the requirement that the interlocking system will prevent a train from moving to a track section that is already occupied.

thisTrain occupies next ran(occupies)

For H2, we introduce a guard to setNext that represents the requirement that a point cannot change while it is occupied.

thisTrack ran(occupies)

For H5, we introduce a guard to setOccupies that represents the requirement that the interlocking system will prevent a train from moving to a track section that is failed.

thisTrain occupies next failed = FALSE

For H6, we introduce a guard to setOccupies that represents the requirement that the interlocking system will prevent a train from moving to a track section that is 'incorrect'.

thisTrain occupies next incorrect = FALSE

For H6, we also introduce a guard to setIncorrect that represents the requirement that the interlocking system will prevent a track section from becoming incorrect while it is occupied.

thisTrack ran(occupies) boolVal = FALSE

For this stage of the model, the Rodin tools generated 28 proof obligations, all of which were discharged automatically.

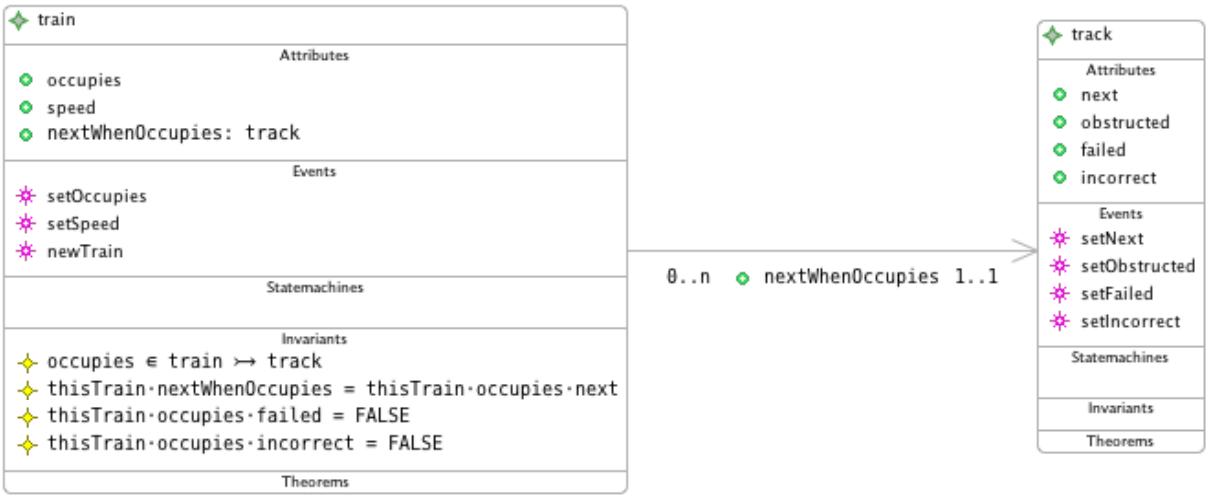


Figure 5.7 - The final refinement consists of the two refined classes with an additional association to represent the value of next when a track is first occupied by a train. The four safety invariants have been added to the class train.

The hazards make no mention of trains failing to stop at the end of a line. Hence we assumed in our model that this is not a safety requirement. Of course, this could be an omission from the hazards but because our model faithfully recreates the given hazards in this respect, none of the verification tools

detect that there is a problem. Although the verification tools can detect many internal consistency problems, they cannot decide whether the model is what the users want. Validation of the model could be addressed by animating the model in the presence of domain experts using the Rodin animation tools. (There are two animation plug-ins for Rodin, ProB and AnimB.) If the animation reveals that trains are supposed to stop when there is no next track to go to, suitable guards and invariants could be added to prevent this hazard.

5.4 Summary of Achievements

We briefly summarise what has been achieved with this model.

The model is intended as an illustration. When the informal safety requirements have been finalised, this model may require revision to make it more realistic. For example the model does not consider the possibility of track being bi-directional. Other formalisations of the safety requirements may also be required in other formal notations.

The model illustrates how some example hazards can be used to derive safety requirements. To derive safety requirements it was first necessary to introduce some concepts that are present in the domain. We also made explicit which hazards are not covered by an interlocking system. The four invariants given in Figure 5.8 are a formalisation of the relevant safety requirements in terms of these domain concepts.

In addition, we introduced some very abstract behaviour and proved that this behaviour satisfies the formalised safety requirements. By modelling the safety requirements and a behaviour that is safe, and proving this to be true, we gain confidence that the safety requirements are consistent. By keeping the model simple and abstract we gain confidence that the safety requirements are correct. We should also validate the model by animation.

Using refinement (according to the Rodin tools upon which UML-B is based) it would be possible to make this abstract model more detailed until it describes an interlocking system. The Rodin tools force us to prove that the more detailed model is a proper refinement of this abstract one and hence that the interlocking system also satisfies these safety requirements.

Section 6 – CONCLUSIONS

In this deliverable D.D.4.4 associated with Task D.4.4, we presented our methodology for producing safety requirements in formal format, and we gave an example of how a small set of safety requirements can be formalised in our approach.

The first part of this task is to identify a suitable set of safety requirements for formalisation. The safety requirements of the CENELEC Phase 3 Hazard List are too high-level for this purpose. In order to obtain safety requirements, we have carried out a study of the relevant academic literature and consulted with the railways. Moreover, we have studied a relevant document stemming from the Euro-Interlocking project. Although the requirements obtained from the latter document are for an early version of the Interlocking Model that lacks much of the functionality provided by current or future Interlocking Models, the level of detail in those requirements is more appropriate for our purposes than the CENELEC Phase 3 Hazard List.

We point out that this document does not contain a list of fully formalised safety requirements ready to be input to the verification tools due to the following:

For most of the verification tools, the actual formalisation of safety requirements depends on the availability of the formal Interlocking Model, and the Interlocking Model will only be available in Month 28 of the project.

Due to the above, and in accordance with the Technical Amendment Report to the Description of Work regarding this deliverable, we have focused on presenting the methods we will use to produce formal safety requirements. We have illustrated our approach by presenting a formalisation in UML-B of a small set of requirements put together in the context of the Euro-Interlocking project. UML-B will serve as our common specification language, meaning that any obtained safety requirements will first be stated in a precise and unambiguous manner using UML-B. This already allows some verification using the Rodin toolset. Once the formal Interlocking Model is available, the safety requirements collected in this document can be aligned with this model. The translation of the safety requirements from UML-B into the property languages of the other verification tools (e.g., FDR2, mCRL2, etc.) is straightforward.

Section 7 – BIBLIOGRAPHY

(2006). Rigorous Development of Complex Fault-Tolerant Systems [FP6 IST-511599 RODIN project]. RODIN Book, Springer.

Abrial, J.-R. (2006). Train Systems. RODIN Book, Springer.

Björner, D. (2000). Formal Software Techniques in Railway Systems. 9th IFAC Symposium on Control in Transportation Systems, Germany, VDI/VDE-Gesellschaft Mess- und Automatisieringstechnik, VDI-Gesellschaft für Fahrzeug- und Verkehrstechnik.

Blanco, R. (2001). Using Formal Methods to Validate Geographic Data for a Railway Signalling System.

Bozga, M., J.-C. Fernandez, et al. (2000). "Verification and test generation for the SSCOP protocol." Sci. Comput. Program. **36**(1): 27-52.

Cavalcanti, A., A. Sampaio, et al. (2003). "A Refinement Strategy for Circus." Formal Asp. Comput. **15**(2-3): 146-181.

Cimatti, A., F. Giunchiglia, et al. (1998). "Formal Verification of a Railway Interlocking System using Model Checking." Formal Asp. Comput. **10**(4): 361-380.

Cimatti, A., P. L. Pieraccini, et al. (1999). Formal Specification and Validation of a Vital Communication Protocol. FM'99 - Formal Methods, World Congress on Formal Methods in the Development of Computing Systems, Toulouse, France, September 20-24, 1999, Proceedings, Volume II, Springer.

Eriksson, L.-H. (1996). Specifying Railway Interlocking Requirements for Practical Use. Proceedings of the 15th International Conference on Computer Safety, Reliability and Security (SAFECOMP'96), Springer-Verlag.

Faulkner, A. (2004). Data Integrity: An often-ignored aspect of safety systems, Engineering Doctoral Thesis, University of Warwick.

Fokkink, W. (1996). Safety criteria for the vital processor interlocking at Hoorn-Kersenboogerd. 5th Conference on Computers in Railways (COMPRAIL'96), Volume I: Railway Systems and Management.

Freitas, L., J. Woodcock, et al. (2006). "State-rich model checking." ISSE **2**(1): 49-64.

Gnesi, S., D. Latella, et al. (2000). An Automatic SPIN Validation of a Safety Critical Railway Control System. 2000 International Conference on Dependable Systems and Networks (DSN 2000) (formerly FTCS-30 and DCCA-8), 25-28 June 2000, New York, NY, USA, IEEE Computer Society.

Groote, J. F., J. W. C. Koom, et al. (1994). The safety guaranteeing system at station Hoom-Kersenboogerd, Logic Group, Preprint Series, Department of Philosophy, Utrecht University.

Hall, A. (1990). "Seven Myths of Formal Methods." IEEE Software **7**(5).

Hansen, K. M. (1996). Linking Safety Analysis to Safety Requirements - Exemplified by Railway Interlocking Systems, Department of Information Technology, Technical University of Denmark.

Hansen, K. M. (1996). "Modelling railway interlocking systems."

Hansen, K. M., A. P. Ravn, et al. (1998). "From Safety Analysis to Software Requirements." IEEE Trans. Software Eng. **24**(7): 573-584.

Haxthausen, A. E. and J. Peleska (2000). "Formal Development and Verification of a Distributed Railway Control System." IEEE Trans. Software Eng. **26**(8): 687-701.

Hlavaty, T., L. Preucil, et al. (2001). "Case Study: Formal Specification and Verification of Railway Interlocking System." EUROMICRO Conference.

Huber, M. (2001). Towards an Industrial Applicable Model Checker for Railway Signalling Data, University of York Department of Computer Science, MSc SCSE Project Dissertation.

Ingleby, M. and D. J. Mee (1995). A calculus of hazard for railway signalling. Workshop on Industrial-Strength Formal Specification Techniques, Boca Raton, Florida.

King, T. (1994). Formalising British Rail's Signalling Rules. FME '94: Industrial Benefit of Formal Methods, Second International Symposium of Formal Methods Europe, Barcelona, Spain, October 24-18, 1994, Proceedings, Springer.

King, T. (1994). Logical Elements of Interlocking Systems – Draft for Academic Use, Railtrack plc.

Morley, M. J. (1993). Safety in Railway Signalling Data: A Behavioural Analysis. 6th annual Workshop on Higher Order Logic Theorem Proving and its Applications, Vancouver, 4-6 August, 1993, Springer.

Morley, M. J. (1996). Safety Assurance in Interlocking Design, University of Edinburgh.

Petersen, J. L. (1998). Automatic verification of railway interlocking systems: a case study. Proceedings of the Second Workshop on Formal Methods in Software Practice, March 4-5, 1998, Clearwater Beach, Florida, ACM.

Raili, E. L. (1996). The Verification of the Design of Railway Networks, University of York, Department of Computer Science, MSc SCSE Project Dissertation.

Robinson, N. and G. Nikandros (2003). Railway Signalling Design Tools - Supporting Control Table Designers. IRSE Aspect 2003 Signalling Conference, Queen Elizabeth Conference Centre, Westminster, London.

Schacher, M. (2009). Mini Interlocking - Hazards and Requirements. UIC extranet.

Simpson, A., J. Woodcock, et al. (1997). The mechanical verification of solid-state interlocking geographic data. Formal Methods Pacific ’97, Springer.

Simpson, A. C. (1998). Model Checking for Interlocking Safety. Proceedings of the Second {FMERail} Seminar, 15--16 October 1998, London.

Winter, K. (2002). Model Checking Railway Interlocking Systems. Computer Science 2002, Twenty-Fifth Australasian Computer Science Conference (ACSC2002), Monash University, Melbourne, Victoria, January/February 2002, Australian Computer Society.

Winter, K., W. Johnston, et al. (2006). Tool support for checking railway interlocking designs. SCS '05: Proceedings of the 10th Australian workshop on Safety critical systems and software, Darlinghurst, Australia, Australian Computer Society, Inc.

Winter, K. and N. Robinson (2003). Modelling large railway interlockings and model checking small ones. ACSC '03: Proceedings of the 26th Australasian computer science conference, Darlinghurst, Australia, Australian Computer Society, Inc.