Ensuring Safety with System Level Formal Modelling

Thierry Lecomte(✉), Mathieu Comptier, Julien Molinero, and Denis Sabatier

ClearSy, 320 avenue Archiméde, Aix en Provence, France
thierry.lecomte@clearsy.com

Abstract. During the last five years, Event-B formal modelling has been successfully applied to various railway systems to demonstrate safety early in the design process or once systems are in operation. This approach is aimed at formalising a safety reasoning instead of modelling every bit of the system. This approach is intrinsically fit to scale up to large systems (or system of systems), hence able to handle centralised or distributed systems.

Keywords: B method · Safety platform · Automated proof

1 Introduction

Railway signalling systems, legacy, new, and/or forthcoming, are complex systems, difficult to validate and certify. A large number of works have been reported over years, taking into account exploitation procedure [7], interlocking design data [6], hybrid systems [10], distributed systems [4,5]. Some projects report difficulties to scale up or to properly transmit knowledge through the very formal models.

Since several years, CLEARSY has driven large projects about using formal proofs at system level for railway signalling systems. The fundamental goal in these projects is, instead of modelling all the components of a signalling system, to extract the rigorous reasoning that establishes that the considered system ensures its requested properties, and to assert that this reasoning is correct and fully expressed. The concerned systems were either under preliminary specification, under design or already existing.

This paper makes clear the recent advances in the domain of system-level modelling aimed at demonstrating the safety of railways signalling systems, that remains manageable by the signalling engineer and understandable by the recipient of the study.

This paper is structured in six parts. Section 2 introduces the Terminology. Section 3 briefly introduces the B method. Section 4 presents the methodological framework. Section 5 presents some applications on real signalling systems performed the last years.

T. Margaria and B. Steffen (Eds.): ISoLA 2020, LNCS 12478, pp. 393–403, 2020.
https://doi.org/10.1007/978-3-030-61467-6_25

2 Terminology

This section contains specific definitions, concepts, and abbreviations used throughout this paper.

Atelier B is an Integrated development environment (IDE) supporting the B method and the B language for software development, and Event-B for system-level analysis.

B0 is a subset of the B language that must be used at implementation level. It contains deterministic substitutions and concrete types. B0 definition depends on the target hardware associated to a code generator.

Safety refers to the control of recognized hazards in order to achieve an acceptable level of risk.

SIL put for Safety Integrity Level [11], is a relative level of risk-reduction provided by a safety function. Its range is usually between 0 and 4, SIL4 being the most dependable and used for situations where people could die.

3 Introduction to the B Method

B [1] is a method for specifying, designing, and coding software systems. It covers central aspects of the software life cycle (Fig. 1): the writing of the technical specification, the design by successive refinement steps and model decomposition (layered architecture), and the source code generation.

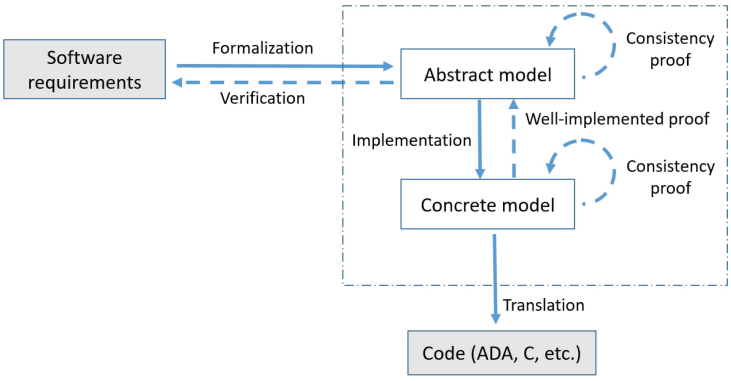

Fig. 1. A typical B development cycle, from requirements to code.

B is also a modelling language that is used for both specification, refinement (Fig. 2), and implementation. It relies on substitution calculus, first order logic and set theory. All modelling activities are covered by mathematical proof that finally ensures that the software system is correct.

B is structured with modules and refinements. A module is used to break down a large software into smaller parts. A module has a specification (called a

machine) where are formalized both a static and a dynamic description of the requirements. It defines a mathematical model of the subsystem concerned:

- an abstract description of its state space and possible initial states,
- an abstract description of operations to query or modify the state.

This model establishes the external interface for that module: every implementation will conform to this specification. Conformance is assured by proof during the formal development process. A module specification is refined. It is re-expressed with more information: adding some requirements, refining abstract notions with more concrete notions, getting to implementable code level. Data refinement consists in introducing new variables to represent the state variables for the refined component, with their linking invariant. Algorithmic refinement consists in transforming the operations for the refined component. A refinement may also be refined. The final refinement of a refinement column is called the implementation, it contains only B0-compliant models. In a component (machine, refinement, or implementation), sets, constants, and variables define the state space while the invariant define the static properties for its state variables. The initialisation phase (for the state variables) and the operations (for querying or modifying the state) define the way variables are modified. From these, proof obligations are computed such as: the static properties are consistent, they are established by the initialisation, and they are preserved by all the operations. Atelier B contains a model editor merging model and proof (Fig. 3) by displaying the number of proof obligations associated to any line of a B model, its current proof status (fully proved or not) and the body of the related proof obligations. An Event-B project is a project containing one refinement column (one specification machine refined several times) and possibly context machines (machines containing only constants and sets definitions). Instead of B operations called sequentially, Event-B components contains atomic events which may be triggered when their enabling condition holds. An Event-B model represents the specification of a system (a device, a procedure, a business rule, etc.) with asynchronous transitions from one state to an other. Similarly to B models,

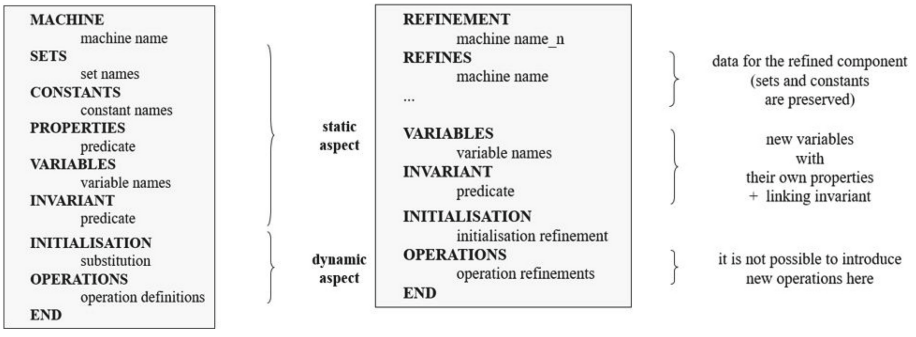

Fig. 2. Structure of MACHINE and REFINEMENT components.

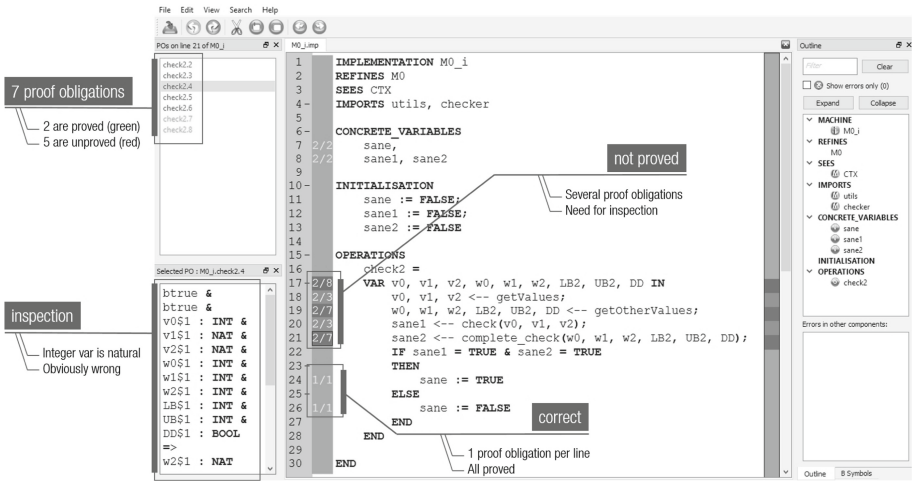

Fig. 3. Atelier B model editor showing proof status.

Event-B models have to verify their invariant properties: initialisation should establish invariant, and for each event fired, if the invariant was true before then it remains true after the execution of the event.

4 Methodology

The methodology described is issued from [2,3,8]. Figure 4 illustrates its different stages, which can be called "the ideal formal world" and which makes it possible to obtain a system that is guaranteed to be zero-defect.

The left side of the diagram represents the "formal proof of correct interoperability". The aim is to ensure that if the individual sub-systems making up the overall solution are implemented in accordance with their specifications, then the safety of the overall system is guaranteed. This proof enables the entity responsible for the integrated system to ensure that there are no hidden safety bugs in the subsystem breakdown.

The right side of the schema could be named "formal proof of correct design". It is a question of guaranteeing that a given implementation (RBC for example) is designed in such a way that the safety expectations expressed in the specifications are effectively met. This part is not included in the project as it is not required and is only provided for illustration.

The "formal proof of correct interoperability" is in three steps, detailed below.

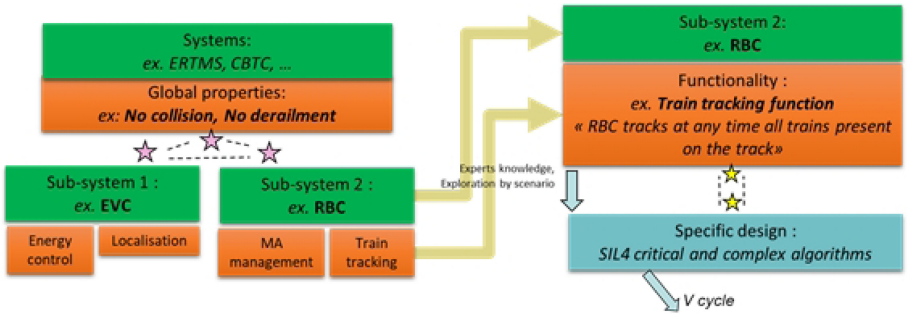

Fig. 4. The complete picture of the formal approach for safe systems.

4.1 Overall Safety Study, Roles and Responsibilities of the Subsystems

The objective is to acquire sufficient knowledge of the selected architecture and of the functional decomposition planned (or anticipated) within the framework of the target project.

It allows more concretely to identify the different actors involved in the security of the global systems as a first analysis of the security principles distributed within these subsystems. This analysis leads to a first version of a set of sub-properties resulting from the breakdown of the global non-collision and non-derailment properties to be respected by the subsystems through their different functionalities. This first phase also enables listing the different functionalities of the subsystems involved in safety and on which the formal proof has to be applied.

4.2 Formal Safety Demonstration Against the Risks of Collision and Derailment

This phase is the core of the formal proof of correct interoperability. It comprises a formalisation stage, i.e. a stage where all the elements involved in the safety demonstration (software variables, trains, signal states, points, etc.) are transformed into unambiguous mathematical objects.

4.2.1 Formalizing the Global Properties

The first activity consists in formalizing the global properties (mainly anti-collision and non-derailment) as well as the expected properties of the different subsystems in natural language (English) but with the precision of the mathematical language, which allows to obtain a definition without any possible ambiguity.

For example, sets can be a possible representation of trains and moving blocks. In this case the anti-collision property can be formalized as follows

$$\forall\, t_1,\, t_2 \in \mathit{trains}_{id} \Rightarrow t_1 \cap t_2 = \emptyset$$

where t_1 and t_2 represent the locations covered by each train.They are typed as sets. The anti-collision property implies that their intersection is empty.

Let us imagine that during the first phase of the study, it was identified that safety with regard to anti-collision is based on the principle of blockage, i.e. at any given moment, there is at most one train per block, so the anti-collision property can be refined by this principle of blockage:

$$\forall \boldsymbol{c_i} \in \boldsymbol{blocks_{id}} \Rightarrow (\exists! \, \boldsymbol{t_i} \in \boldsymbol{trains_{id}} \wedge \boldsymbol{t_i} \cap \boldsymbol{c_i} \neq \emptyset)$$

It reads that there exists only one train intersecting with a block (given that, implicitly, blocks do not intersect among themselves).

As long as the blocks do not intersect (reasoning hypothesis taken), it is possible to prove the anti-collision property mathematically from the refined property. In the same way, this property describing the blockage principle can be refined into sub-properties directly applicable to the subsystems.

For example, the ERTMS Blocking Principle will only be possible if the RBC has a correct train tracking (i.e. all trains on the track are tracked by the RBC at all times) but also if its Movement Authority (MA) management function is correct (i.e. the RBC never sends an MA to a train beyond a block already occupied by another train).

4.2.2 Performing an Irrefutable Mathematical Demonstration

The second activity consists of performing an irrefutable mathematical demonstration that the various subsystems meet the expected refined properties. Each functionality of each subsystem is analysed in order to verify that the requirements described in the specifications are sufficient to demonstrate the preservation of the expected properties. This demonstration shall be based on a set of explicit and justified assumptions, i.e. accompanied by the description of worst-case scenarios in case of non-compliance with these assumptions.

For example, the mathematical demonstration of the previous property requiring that a RBC never sends an MA to a train beyond a block already occupied by another train may be based on the assumption that the RBC never extends an MA from a train beyond a restrictive signal.

Assumptions thus obtained will have to be directly described in the specification or will be sub-properties requiring sub-demonstrations based in turn on other assumptions.

The outputs of the steps are:

- A formal safety demonstration (or correct interoperability) with regard to the risks of collision and derailment, described in natural language and formalised in mathematical language.
- The complete list of the so-called safety assumptions: all those used in the previous demonstration, together with their justification (worst-case scenario in case of non-compliance).

4.3 Modelling, Proof, and Animation of Event-B Models with Atelier B and ProB Tools

Atelier B is a tool for system modeling and proof of invariant properties of the modeled system. This modelling is done in a mathematical language: the B language. An "invariant" property is a property that is true at all times. In practice, the aim is to prove the invariance of the negation of the dreaded event (meaning that the opposite situation to the dreaded event is always true, e.g. "no collision").

The proof of invariant properties is based on the principle of induction: there is no evolution of the system leading to the invariant not being respected. The proof is obtained by demonstrating that the initial state of the system conforms to the expected properties, and each of the possible transitions of this same system preserves these properties (assuming the true properties at the input of the function).

The objective here is to model with Atelier B the reasoning conducted in the first two phases described in the two previous paragraphs. The production of these B models, accompanied by formal proof via the Atelier B tool, will ensure that the reasoning carried out on paper does not contain any logical errors and that all the security hypotheses have been expressed (no implicit hypothesis forgotten during the "paper" reasoning).

The outputs are:

- A proof model written in the B language, encapsulating the core concepts and the formal safety reasoning and associated safety concepts;
- A formal proof of important safety properties (no collision, no derailment, etc.) conducted within Atelier-B;
- Various instances of the proof model for animation with ProB, to ensure consistency and functionality. The models will be accompanied by various scenario files, to ensure that the formal models can implement various use cases;
- Vizualisations to ensure that domain experts can inspect the formal model without having to understand the B language (Fig. 5);
- An executable demonstrator for running more extensive functionality tests.

5 Applications

The methodology above have been used at several occasions:

- **Formal proof of the safety properties for the NYCT Line 7 Modernization Project** [9]. The New York City Transit Authority has included formal proofs at system level as part of the safety assessment for its New York subway Line 7 modernization project (Fig. 6), based on the CBTC from Thales Toronto. The main goal was to obtain a formal proof for the main safety properties of the system: no collision and no over-speeding. A book of assumptions was built up, covering every relevant aspect of the system, from

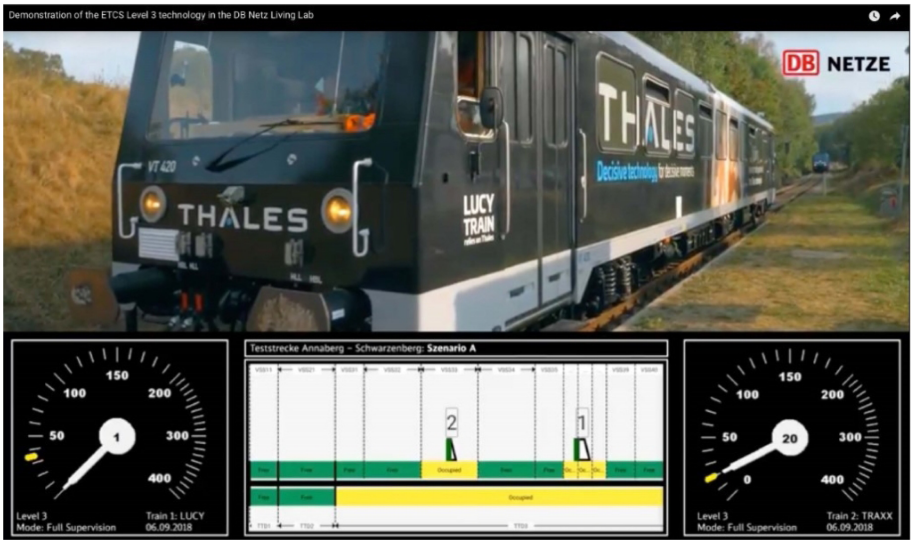

Fig. 5. Formal B model of Hybrid Level 3 Principles running in real-time. Source: Deutsche Bahn, https://www.youtube.com/watch?v=FjKnugbmrP4

internal design to external conditions. Safety properties were then obtained by pure logical reasoning only from these well defined assumptions. This study revealed all assumptions needed and reached a "proof level" confidence for the system properties.

– **Safety Analysis of the Octys CBTC System** [2]. RATP was going to upgrade their subway lines with driver with Octys, with the objective to improve throughput and safety by ensuring continuous train speed control, to participate in ensuring the safety of passenger transfers through the train and platform screen doors, to diminish the headway and to reduce wayside signaling requirements. Octys relies on multi-sourcing and interchangeability. The system is split in different sub-parts that are to be developed by different suppliers and interchangeable in the sense that any compliant subpart, whatever its brand, shall fit seamlessly in the system. Octys had been deployed successively on Paris lines 3, 5 and 9; two other lines were scheduled to be equipped in the near future. The formal safety analysis consisted in expressing properties that are key to the safety of the system both in natural language and mathematical notation, and in constructing formal proofs that these properties hold. The conclusion was that such system level proof is feasible for a system like the Octys CBTC, with the appropriate level of independence with the intricate but out of scope interlocking. The findings (not disclosed) provided their benefits. The output results were expected to be reused as input properties for subsystem formal analysis performed by RATP.

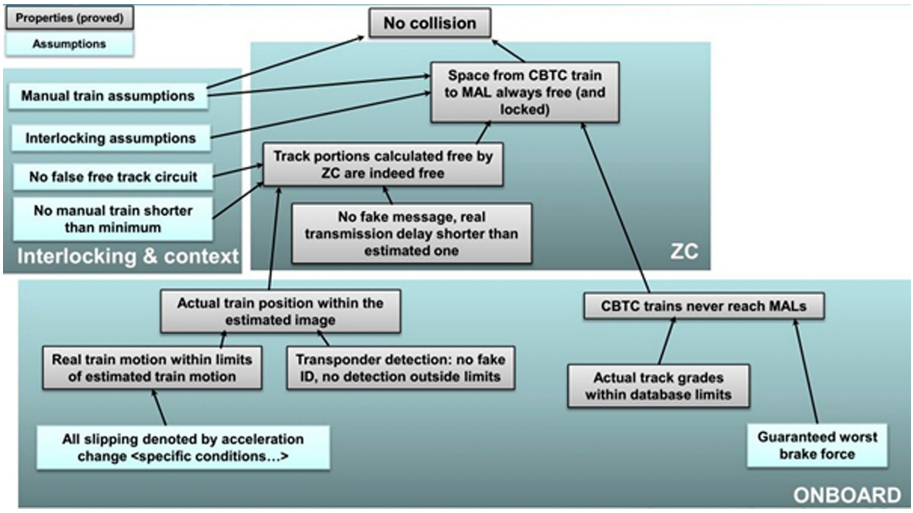

Fig. 6. NYCT line 7 modernization project - the structure of the formal proof for the main safety properties of the system: no collision and no over-speeding.

– **Validation of a CBTC zone controller** [3]. The analyzed CBTC is a flagship product of Alstom that is already in operation for over 95 metro lines worldwide. The formal analysis was applied at the software design level. The objective was to prove that a software specification and its implementation satisfy the expected system properties. The functional part of the analyzed software was developed with the B Method. A formal model of the software components was created, then formally refined and finally formally implemented. With the validation of the high-level specification of each component of a CBTC, we wanted to guarantee system-wide safety properties in light of evolving requirements specification of the components, and taking into account optimization to increase availability of the system. The following results were obtained:

  - Retrieve and/or explain clearly the fundamental design principles.
  - Exhibit and explain formally the assumptions made about the studied function inputs.
  - Retrieve and formalize the historical reasoning of the designers and keep track of their justification.
  - Identify complexity that is not necessary to maintain the properties and has become useless or obsolete, providing opportunities for functional improvements and performance gains.
  - Possibly detect corner cases where the properties are not fulfilled, providing the safety teams the elements necessary to analyze the consequences.
  - Propose design improvements.

  From the point of view of the system teams at the origin of the design, it is no longer necessary to try to imagine all possible combinations of functions.

In return, any newly developed function must preserve the invariant properties. An implemented function is safe as long as it preserves the key invariant properties as exhibited by the study. To validate the safety of an evolution, it is therefore sufficient to require a formal demonstration, and to ensure that it does not contain logical errors. Any evolution can be mathematically proven even before it enters the traditional software development cycle.

6 Conclusion and Perspectives

Formally proving railway signalling systems is a very challenging activity, whether the system is centralized (this is the case of many legacy systems) or distributed. In this paper, we proposed a novel approach. We successfully experimented with the approach on several real-size systems, already in exploitation or to be deployed for the first time. Instead of modelling the different parts of the railway system, the approach extracts the rigorous reasoning that establishes that the considered system ensures its requested properties, and to assert that this reasoning is correct and fully expressed. This approach supports large systems without problem, is able to deliver understandable outputs in natural language as well as graphical model animation to exhibit key scenarios. This approach seems to be adequate for both centralized and distributed signalling systems. Several other experiments are on-going in several European countries for the main lines (ERTMS) that will be reported in the future.

References

1. Abrial, J.: The B-book - Assigning Programs to Meanings. Cambridge University Press, Cambridge (2005)
2. Comptier, M., Déharbe, D., Perez, J., Mussat, L., Pierre, T., Sabatier, D.: Safety analysis of a CBTC system: a rigorous approach with event-B. In: Fantechi, A., Lecomte, T., Romanovsky, A. (eds.) RSSRail 2017. LNCS, vol. 10598, pp. 148–159. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68499-4_10
3. Comptier, M., Leuschel, M., Mejia, L.-F., Perez, J.M., Mutz, M.: Property-based modelling and validation of a CBTC zone controller in Event-B. In: Collart-Dutilleul, S., Lecomte, T., Romanovsky, A. (eds.) RSSRail 2019. LNCS, vol. 11495, pp. 202–212. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-18744-6_13
4. Geisler, S., Haxthausen, A.: Stepwise development and model checking of a distributed interlocking system using raise. Formal Aspects Comput. (2020)
5. Hei, X., Takahashi, S., Nakamura, H.: Distributed interlocking system and its safety verification, pp. 8612–8615 (2006)
6. Iliasov, A., Stankaitis, P., Adjepon-Yamoah, D.: Static verification of railway schema and interlocking design data. In: Lecomte, T., Pinger, R., Romanovsky, A. (eds.) RSSRail 2016. LNCS, vol. 9707, pp. 123–133. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33951-1_9
7. Metayer, C., Clabaut, M.: DIR 41 case study. In: Börger, E., Butler, M., Bowen, J.P., Boca, P. (eds.) ABZ 2008. LNCS, vol. 5238, pp. 357–357. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-87603-8_44

8. Sabatier, D.: Using formal proof and B method at system level for industrial projects. In: Lecomte, T., Pinger, R., Romanovsky, A. (eds.) RSSRail 2016. LNCS, vol. 9707, pp. 20–31. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33951-1_2
9. Sabatier, D., Burdy, L., Requet, A., Guéry, J.: Formal proofs for the NYCT line 7 (flushing) modernization project. In: Derrick, J., Fitzgerald, J., Gnesi, S., Khurshid, S., Leuschel, M., Reeves, S., Riccobene, E. (eds.) ABZ 2012. LNCS, vol. 7316, pp. 369–372. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30885-7_34
10. Stankaitis, P., Iliasov, A.: Theories, techniques and tools for engineering heterogeneous railway networks. In: Fantechi, A., Lecomte, T., Romanovsky, A. (eds.) RSSRail 2017. LNCS, vol. 10598, pp. 241–250. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68499-4_16
11. Wikipedia contributors: Safety integrity level - Wikipedia, the free encyclopedia (2020). https://en.wikipedia.org/wiki/Safety_integrity_level. Accessed 08 May 2020