# B-RAIL : Risk analysis and specification

Jean-Louis Boulanger

# B-RAIL : Risk analysis and specification

Jean-Louis Boulanger

Université de Technologie de Compiègne,
Address: Laboratoire Heudiasyc, UMR CNRS 6599, 60205 Compiègne Cedex, France.
Phone: (+33) 3 44 23 44 23, Fax: (+33) 3 44 23 44 77
e-mail: jean-louis.boulanger@utc.fr

**Abstract**: In the European railways standards (CENELEC EN 50126 [4], EN 50128 [5], EN 50129 [6]), it is required to obtain evidence of safety in system requirements specifications. The focus of this paper is on the development of system requirements specifications with respect to fulfilling demands of European railways standards. In spite of progress carried out in software development, designing a complex system while respecting its safety requirements, remains very hard. Ambiguities and defects in system requirements specification may have consequences on the whole system development.

**Keywords**: Formalization, Level Crossing, Risk analysis, System Requirements, Traceability, UML.

## 1. Introduction

In the railway domain, safety requirements are obviously severe. It is very important to keep requirements traceability during software development process even if the different used models are informal, semi formal or formal. We investigate how the Unified Modelling Language (UML), can be used to formally specify and verify critical railways systems. A benefit of using UML is it status as an international standard (OMG) and its widespread use in the software industries.

Safety invariants can be derived from hazard analysis and can be supported by a system model in diagrams of UML.

In this paper, we purpose a method for modelling a safety railways application. But the precondition to use UML diagrams for system specification, which is usable for formal correctness proofs and refutation checks, is that the UML has to be used with a precise semantics. This is possible by definitions of translation rules for the conversion of UML notation in a formal language. This study is integrated into a larger one (called B-RAIL) that aims at linking an informal approach (UML notation) to a formal (B method) one.

## 2. UML

Born from the different object methods, like OMT or Booch & Jacobson, and normalised by the Object Management Group[1], Unified Modelling Language (UML) has now become a standard to model systems. The UML notation [11] makes it possible to model an application according to an object view. Many different diagram types make this model. Each diagram allows a particular view of the system.

The 9 more important diagrams are:

- Use Case diagram,
- Component Diagram,
- Collaboration Diagram,
- Class Diagram,
- Deployment Diagram,
- State Diagram,
- Activity Diagram,
- Sequence Diagram,
- Object Diagram.

The reader interested by more details in syntactic and semantic aspects can refer to the reference guide of UML [10]. Even if UML notation is a language in which models can be represented, it doesn't define the making process of these models. Nevertheless, several dedicated tools have strengthened the popularity of UML. These tools allow graphic notation and partial generation of the associated code and documentations. The UML notation is known by most computer scientists and is now used in several domains. Using UML class diagrams to define information structures has now become standard practice in industry. Recently, the critical application domains have used the notation and several questions exist around this use.

In the next part of the paper, among the different possible diagrams, we'll use the state diagrams particularly adapted to reactive system modelling. A state is a condition in an object life while it satisfies some conditions, runs some actions or waits for some events. There are two specials states: initial state and end state. The Initial State is the state of an object before any transition. End States mark the destruction of the object whose state we are modelling. An event is a particular occurrence that can trigger a transition from a state to an other one. A state diagram can represent the system behaviour.

[1] http://www.omg.org

This type of diagram represents finite state automaton, under a graphical representation, linked by oriented arcs describing transitions.

Statechart diagrams ([12], [13]), also referred to as State diagrams, are used to document the various modes ("state") that a class can go through, and the events that cause a state transition. The state-transitions graph formalism is not a UML innovation. It has often been employed in other contexts and a large consensus, from David Harel's works, exists around this notation. It introduces the description of possible sequences of states or actions which can occur to an element during its life. Such sequences arise from element reaction to discrete events.

## 3. Case Study

To illustrate our approach, we will choose to design a level crossing. This example is inspired by Jansen, L. and Schneider, E. [9]. The term level crossing, in general a crossing at the same level, i.e. without bridge or tunnel, is especially used in the case where a road crosses a railway; it also applies when a light rail line with separate right-of-way crosses a road; the term "metro" usually means by definition that there are no level crossings.

Firstly, a single-track line, which crosses a road in the same level, is modelled (figure 1).

The crossing zone is named danger zone. The most important security rule is to avoid collision by prohibiting road and railway traffic simultaneously on level crossing. The railway crossing is equipped with barriers and road traffic lights to forbid the car passage.

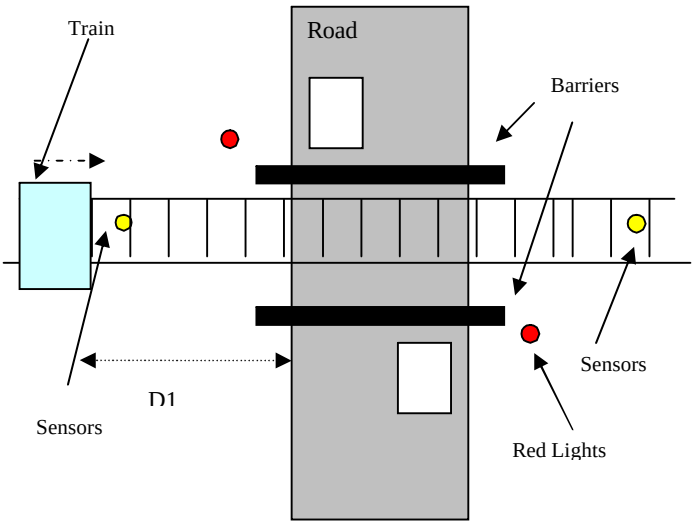


Figure. 1- Single-track line level crossing

Two sensors appear on the railroad to detect the beginning (train entrance) and the end (train exit) of the level crossing protection procedure. The level crossing is not in an urban zone; this implies a sound signalization.

Traffic lights consist of two lights: one red and one yellow. When they are switched off, road users (drivers, cyclists, pedestrians,…) can cross.

When the yellow light is shown road shall stop at the level crossing if possible. In the other case, the level crossing is closed and railway traffic has priority. The yellow and red light never must be shown together.

## 3. Requirement

3.1 Environment
It is often difficult to understand requirements if they are stated as a list. For that reason, functional requirements (and even some non-functional requirements) can be expressed by using some "use cases".

A use case analysis involves the following steps:

- Determine the actors, i.e. any outside entities (people, systems, etc.) that interact with the system.
- Identification of Use Cases (name, purpose, goal, pre- and post-condition, ..).

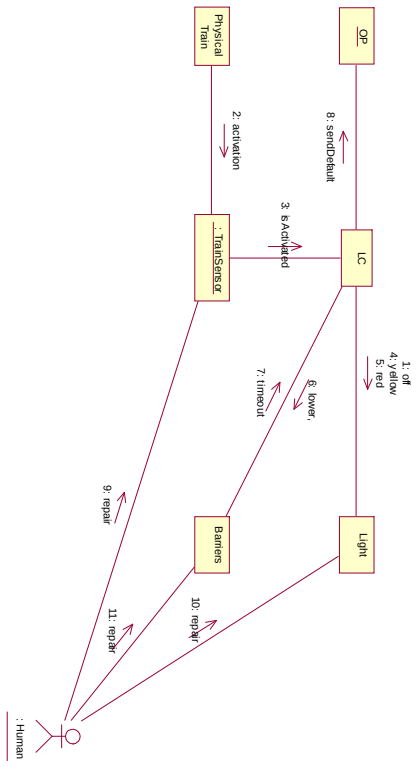


Figure. 2 – Contextual diagram

A use case diagram describes and traces the functional requirements of the system and describe how the system can and will be used. The use case diagram gives an overview of the model.

***UR3**: The railway crossing is equipped with barriers and road traffic lights. Traffic lights at the level crossing consist of a red and a yellow light.*

3.2 Failures

The user requirement gives information concerning the failures and their direct effects on the system.

***UR12:** Possible failure conditions have to be taken into account for a safe control of the level crossing and the train.*

In our model, failures of yellow or red traffic lights (to be separately), barriers, the vehicle sensor and the delay or loss of radio network are considered.

Use case of figure 3 is an example where we model some communication failures.

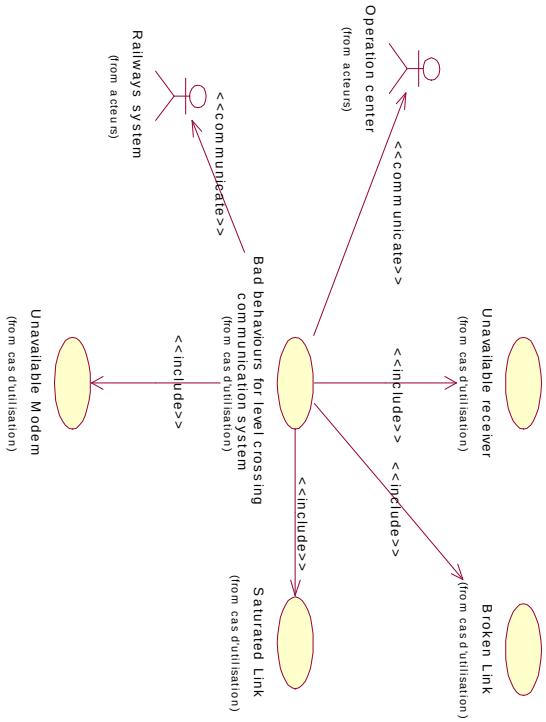


Figure. 3 – Use case

Operational scenarios can be specified by means of sequence diagrams of UML (see Figure 4).

3.3 Risk analysis

According to EN 50129 [6], risk analysis essentially consists of four steps:

- system definition;
- identification of operational hazards;
- consequence analysis;
- risks assessments.

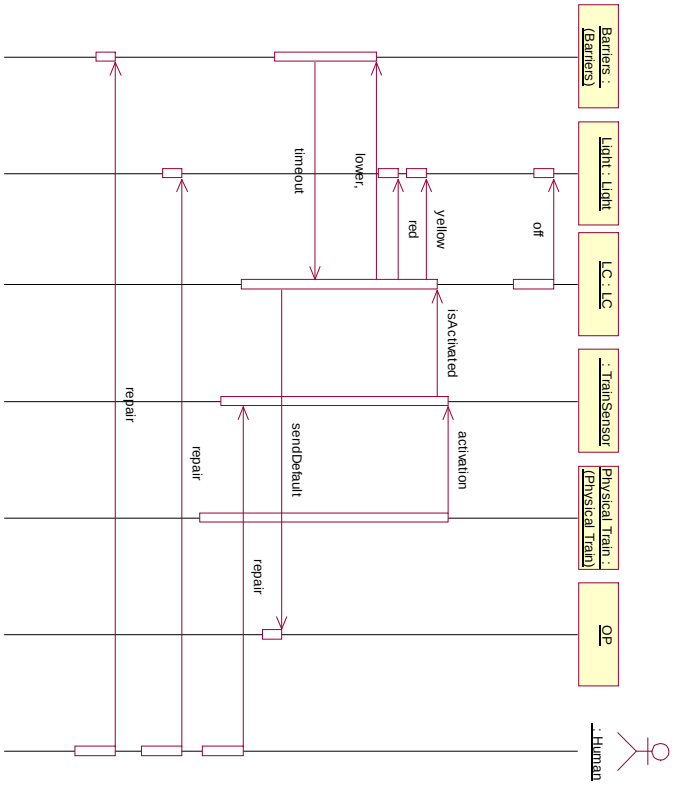


Figure. 4 – Sequence diagram

The identification of operational hazards (step 2) can be done by the analysis of the user requirements (UR) and/or by the analysis of classical risks. In our case, the UR contains:

***UR2:** The intersection area of the road and the railway line is called danger zone, since trains and road traffic must not enter it at the same time to avoid collision.*

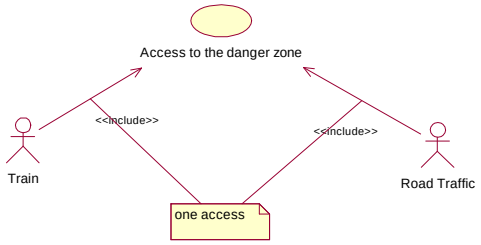


Figure. 5 - Use case from UR2

Figure 5 a use case that described the basic risk.

In first time, we derive safety requirement by using FTA (Fault Tree Analysis). A FTA is a graphical technique that provides a systematic description of the combinations of possible occurrences in a system, which can result in an undesirable outcome (for more information see International standard IEC 61025 [8]). This method can combine hardware failures and human failures. For safety-critical systems, the root node of the tree will often represent a system-wide, catastrophic event taken from a pre-existing hazards list.

From the collision risk we can derive the next FTA:

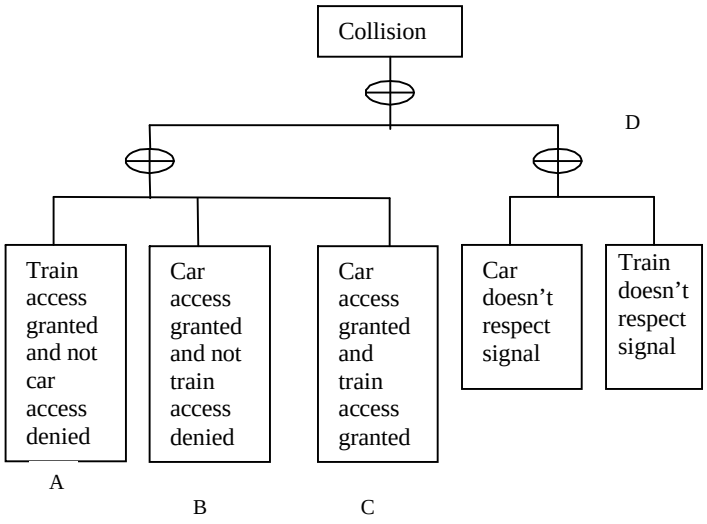


Figure. 6 - Fault Tree Analysis

The first FTA is split in some part. The D part concerns some human errors. The C part introduces the principle property for the system: "The system does not granted access in same time to train and road traffic".

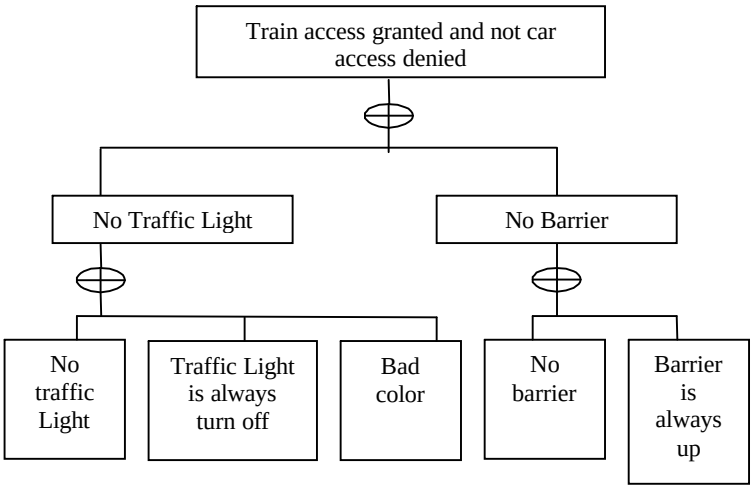


Figure. 7 - Fault Tree Analysis continue

The A and B part deals with absence and failures of equipments (barrier, traffic light, communication, train sensor).

3.4. System Modelling

For modelling the system structure and interfaces between system objects class diagrams are suitable (Figure 8). The class diagram describes the relationships between classes and shows the logical view of a system (static view).

In respect with safety analysis, the control system provides the capability to authorise the danger zone access for road traffic or for train.

This system immediately reports the occurrence and repair of failures to the Operation Centre.

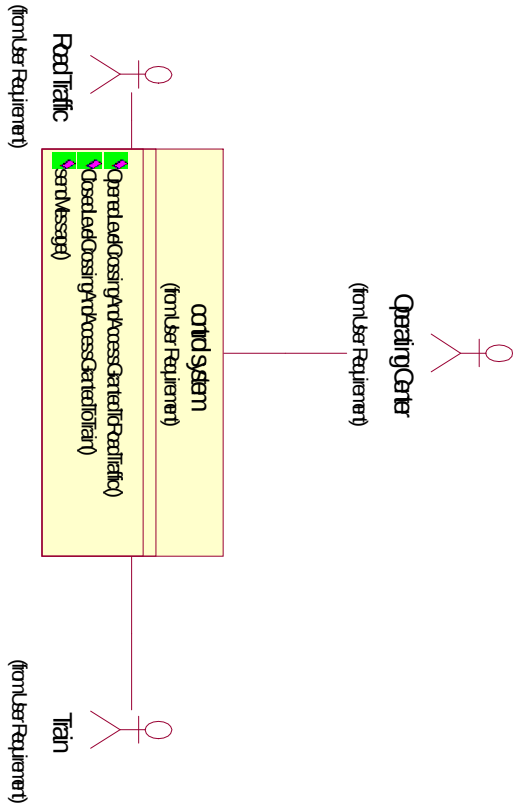


Figure. 8- First Class diagram

3.5 Sub-System Modelling

***UR 3:*** *Decentralized radio-based control system*

This UR indicates that the system is split in 3 parts:

- Communication sub system,
- Train control system (TCS),
- Level crossing system (LCS).

In figure 9, we split the current system in three sub-systems and we introduced some interaction between environment (barriers, ColorLight, physical train and road traffic) with the level crossing application.

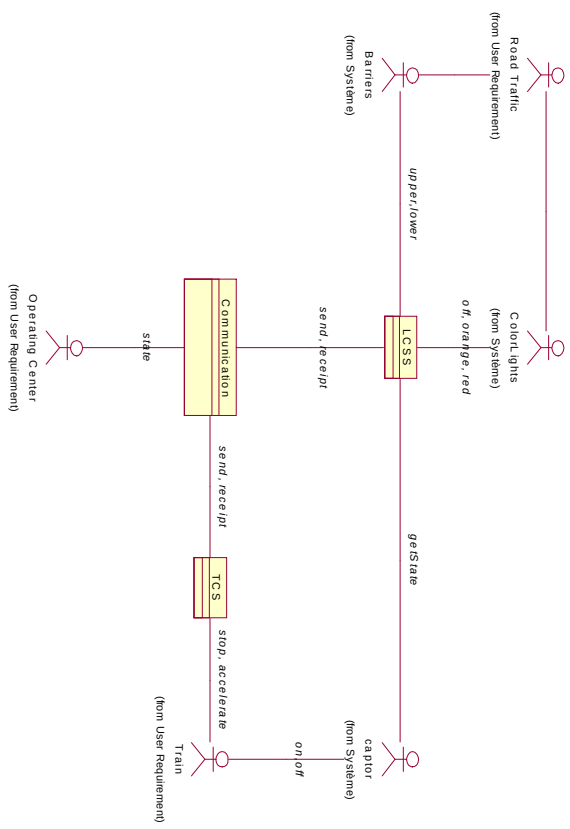


Figure. 9- Sub-system decomposition

The figure 10 purposes a complete class diagram which introduced some interactions between all components (actors, physicals components and applications softwares):

- Level crossing control system and physical equipment (barrier, traffic light, train sensors)
- Train control system and physical equipment (train sensor),
- Level crossing control system and communication,
- Train control system and communication,
- Operation centre and communication

Statechart diagrams (states/transitions diagram), also referred to as State diagrams, are used to document the various modes ("state") that a class can go through, and the events that cause a state transition. The state-transitions graph formalism is not a UML innovation. It has often been employed in other contexts and a large consensus, from David Harel's works, exists around this notation.

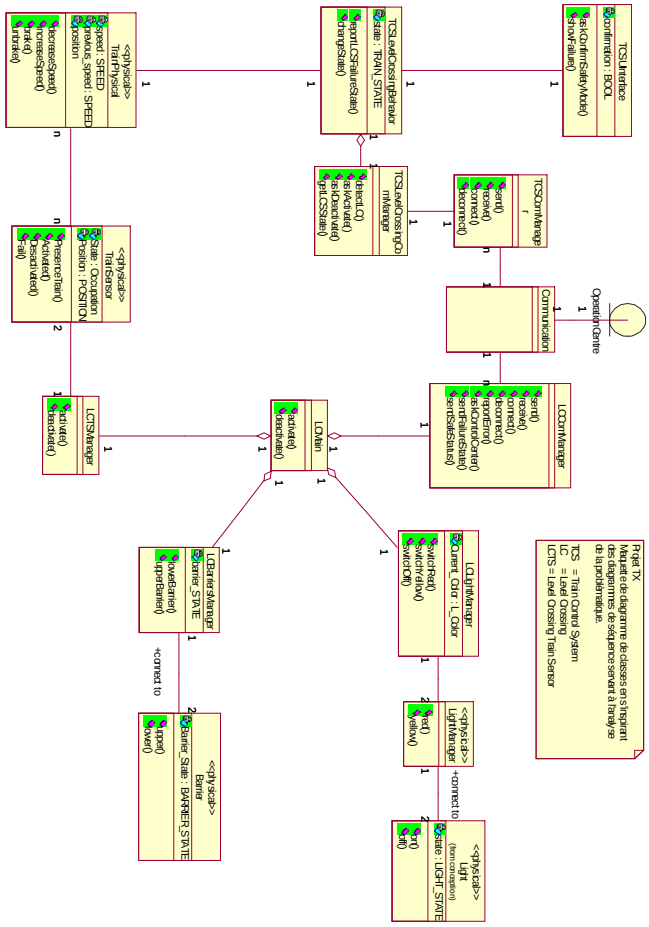

Figure. 10 - Complete class diagram

It introduces the description of possible sequences of states or actions which can occur to an element during its life. Such sequences arise from element reaction to discrete events.

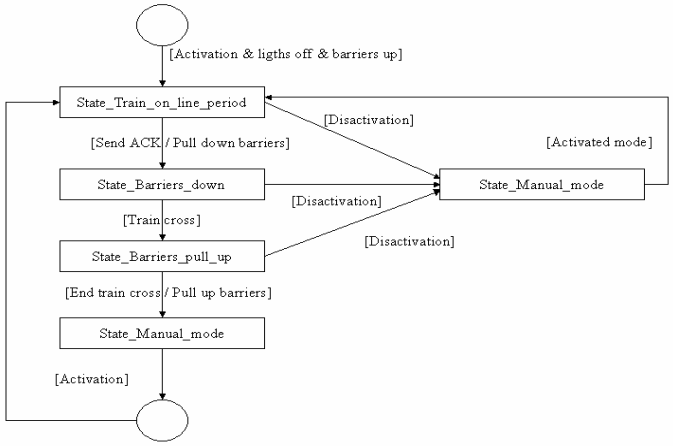


Figure. 11 - Statechart for train behaviour

Figure 11, introduce the behaviour of embedded system. When the train passed the start of danger zone, the embedded system asks to the control system an acknowledgement (ack), the embedded system gets into stand by and begin to brake in order to pull down the barrier in time. After this notice time, the control system sends its state to embedded one. If the level crossing is in safe mode, the embedded system stops the braking and restarts with its initial speed. An end-crossing sensor detects the train exit and starts the barrier pull up and the lights switch off.

We coded all properties in UML by using OCL constraints attached to classes or sets of associations to specify safety and operational invariants of reactive systems in a concise manner.

## 4. Conclusion

The main difficulty to specify railway case study is the less of harmonisation between the different European systems.

The level crossing modelling presented here gives a first step to a computerised management of level crossing.

In this paper, we purpose a method for modelling a safety railways application. But the precondition to use UML diagrams for system specification, which is usable for formal correctness proofs and refutation checks, is that the UML has to be used with a precise semantics. This is possible by definitions of translation rules for the conversion of UML notation in a formal language.

Our global project purposes to transform a semi formal modelling (UML model) to a formal specification (B method, for more information see [1]).

## 5. Acknowledgement

The authors acknowledge the contribution of their colleagues to this work.

## 6. References

[1] Abrial, JR. (1996). ”The B Book - Assigning Programs to Meanings”. Cambridge University Press, August 1996.

[2] Jean-Louis Boulanger, Philippe Bon and Georges Mariano (2004) ”From UML to B - a level crossing case study”, Oral Presentation to COMPRAIL 2004, 17-19 May 2004, Dresden Germany.

[3] Philippe Bon , Jean-Louis Boulanger and Georges Mariano (2003)“Semi formal modelling and formal specification: UML & B in simple railway application”. ICSSEA 2003, December 2-4 2003.

[4] EN 50126, (1999), ”Railways Application – The specification and demonstration of Reliabilty, Availibility, Maintenabiliy and Safety (RAMS)”, 1999.

[5] EN 50128, (2001), ”Railways Application – Communication, signaling and processing systems – Software for railway control and protection systems”, 2001.

[6] prEN 50129, (2000), ”Railways Application – Safety related electronic systems for signaling”, 2000.

[7] Einer, S.; Schrom, H.; Slovák, R.; Schnieder, E. (2002), ”A railway demonstrator model for experimental investigation of integrated specification techniques”, In: Ehrig, H.; Grosse-Rhode, M., Hrsg.: ETAPS 2002 - Integration of Software Specification Techniques, S. 84-93, TU Berlin, DFG, Grenoble 2002.

[8] International standard IEC 61025 (1990), ”Fault Tree Analysis (FTA)”, International Electrotechnical Commission, Geneva, Suisse, 1990.

[9] Jansen, L. and Schneider, E. (2000), « Traffic Control Systems Case Study: Problem Description and a Note on Domain-Based Software Specification », Institute of Control and Automation Engineering, Technical UNIVERSITY of Braunschweig, 2000.

[10] OMG (2004), “Unified Modelling Language version 2.0 ”, report 2004.

[11] MULLER P.-A., "Modélisation objet avec UML", Éditions Eyrolles, 2001.

[12] David Harel, et al. “On the Formal Semantics of Statecharts.”, Proceedings of the 2nd IEEE Symposium on Logic in Computer Science, IEEE Press, NY, 1987. pp. 54–64.

[13] David Harel, “On Visual Formalisms.” Communications of the ACM, vol. 31, no. 5, May 1988. pp. 514–530.

## 7. Glossary

*FTA* Fault Tree Analysis

*LCS* Level crossing system

*TCS* Train control system

*UML* Unified Modeling language

*UR* User Requirement