Process Model for the Development of System Requirements Specifications for Railway Systems

Dipl.-Ing. Friedemann Bitsch[1]
Institute of Industrial Automation and Software Engineering (IAS),
University of Stuttgart, Germany
Pfaffenwaldring 47
D-70550 Stuttgart
Tel.: +49 (0)711 685-7292
Fax: +49 (0)711 685-7302
E-mail: bitsch@ias.uni-stuttgart.de

Abstract: In this paper a process model for the development of system requirements specifications for railway systems is introduced. Demands of the approval of system requirements specifications, which arise from recent European railway standards, are taken into account. The aim is to obtain a system specification, which is unambiguous and easy to understand for all parties involved and in which safety aspects are considered in detail. Correlations between the development of a precise system specification, the performance of safety relevant correctness checks and the performance of risk analysis are presented. Especially the identification, specification and formalisation of safety requirements are treated with regard to correctness checks referred to safety aspects by using model checking. It is also demonstrated how different techniques of risk analysis can be supported by a system model in diagrams of the Unified Modelling Language (UML). This work has been developed in close co-operation with the Institute of Railway Systems Engineering and Traffic Safety (IfEV), Technical University of Braunschweig, Germany within the scope of the project *SafeRail*[2].

Key words: system requirements specification, UML, risk analysis, formal specification of safety requirements, evidence of safe functionality.

1. Introduction

Ambiguities and defects in system requirements specifications (also known as ”user requirements specification”) may have consequences on the whole system development. In case of safety critical systems these consequences could be fatal in this way that the system is not safe.

[1] This work was sponsored by the German Research Council (DFG) within the scope of the focus area program (1064) on the “Integration of Specification Techniques with Applications in Engineering”.
[2] Project of the German Research Council (DFG) within the scope of the focus area program (1064) on the “Integration of Specification Techniques with Applications in Engineering” (see http://www.uni-stuttgart.de/projekte/saferail/).

For those reasons in the new European railway standards (CENELEC [EN50126], [EN50128] and [prEN50129]) and in the international generic standard for safety related systems [IEC61508] it is required to obtain evidence of safety in system requirements specifications. Moreover, the approval of the system requirements specification by an independent supervising authority is demanded. More and more the fulfilment of CENELEC standards is required by European requests for proposal. The problem is that these standards often prescribe what should be done but they only contain little guidance on how the demands can be realised in concrete projects (see [No01]). In this first system development stage different techniques and methods must be used to define the system and to pursue safety strategies. With the goal to reach an homogenous and consistent system requirements specification it is necessary to clarify the syntax and semantics of the used specification and modelling techniques. There also has to be checking possibilities ensuring correctness related to safety requirements. Also the logical interrelationship between the different techniques and how they support each other has to be clear.

The focus of this paper is on the development of system requirements specifications with respect to fulfilling demands of standards. Methods, techniques and approaches, which are integrated in a methodical way in form of a process model for the development of system requirements specifications for railway systems, are introduced. In such process model, demands not only for standards have to be fulfilled but also for the usability and understandability for the different parties involved at the development of system requirements specifications for safety critical application fields. The first party involved are the *operating authorities*, who are the clients. In Germany, in railway practice, this is the “Deutsche Bahn” (DB). Its duty, according to standards, is the preparation of the system requirements specification. But often they pass the execution of this task on industrial *developers (manufacturers, suppliers)* or they do it in co-operation with industrial developers, who are the second party involved. The third party is formed by the *supervising authorities* (*safety authorities*), who have to inspect the system requirements specification, ensuring the compliance with the standards. In Germany e.g. this is the “Eisenbahnbundesamt” (EBA). But they often pass the inspection on a fourth party involved, the *assessors* who are independent experts in the application area. After the acceptance of the system requirements specification, it is given to industrial *developers*, who will realise the system requirements specification in the role of the supplier. But before, they have to understand the user requirements. The correct understanding is checked this way so that the developers formulate more precisely the requirements in form of the system design specification (target specification). On that basis the operating authority is able to check whether the developers have understood correctly the user requirements. The conclusion is that at the development of system requirements specifications safety does not only have to be achieved, but it also has to be demonstrated for the supervising authority. Such demonstration is the basis for the approval of system requirements specifications. That means that it has to be clear for inspectors of the supervising authority how safety is achieved. Therefore methods and techniques in use have to be understandable for inspectors.

In section 2, demands of European standards in railway area are explained in relation to required techniques and methods. It is also shown a way on how requirements can be fulfilled by using several specific techniques and their combinations. In the next three sections certain methods and techniques, which are used in the DFG project *SafeRail* in perspective to requirements of the standards, are described in more detail: First, in section 3 the precise use of diagrams of UML (Unified Modelling Language) is discussed. Second, in section 4 risk analysis by using FMEA (Failure Mode and Effect Analysis), FTA (Fault Tree Analysis) and ETA (Event Tree Analysis) is explained on the basis of the precise system model in UML notation. Third, in section 5 the formal specification of functional safety requirements is described as basis of safety related correctness and refutation checks of the system model. Finally the paper concludes with the most important results, which lead to an unambiguous and clear system requirements specification.

2. Demands of European Standards in Railway Area

Key messages of [IEC61508] are that absolute safety cannot be achieved, because absolute safety is not an absolute value and therefore safety should be defined as a judgement of the acceptability of the risk for danger. A system is safe if its attendant risks of hazards are judged to be acceptable. Therefore risks, which are not tolerable, have to be reduced. An operational hazard is a situation, in which a threat for humans or the environment exists, caused by the system operation, which could lead to an accident. The risk (harm expectation value) is greater than the marginal risk. This can arise from the system itself or from its environment.

In contrast to previous kinds of system requirements specification, the new CELENEC standards ([EN50126], [EN50128] and [prEN50129]) require the determination and prescription of the safety target for the compound system as part of system requirements specification as well as the safety target for the several subsystems. As a precondition to reach this, the new standards require risk analysis as part of the preparation of the system requirements specification.

In risk analysis possible operational hazards are identified, which have to be considered in the system development by countermeasures. Proceeding from that basis possible consequences of operational hazards are analysed. On that basis tolerable hazard rates can be computed. This way it can be checked if evidence of comparable safety is achieved. “Comparable safety” is the safety of existing systems, which are in operation. In this way quantitative values of risk can be prescribed. In the further steps of system development it has to be considered that these values are not exceeded.

Besides, in standards the use of formal methods for evidence of safe functionality is highly recommended for high safety demands. The advantage of this is that formal models are specified in notations with unambiguous syntax and semantics so that mathematical processes on the basis of such specifications are possible. The result are unambiguous and compact specifications, which even enable correctness proofs and refutation checks concerning functional safety requirements in order to obtain evidence of safe functionality. Therefore the standards recommend the use of formal methods at least for the software parts of system specification in case of high safety critical systems, but it is highly recommended for the highest Safety Integrity Level. However from the point of view of the German supervising authority (Eisenbahnbundesamt) there is the objective of using formal methods for compound system aspects (see [KaLeSu00]). In addition a formal modelling of the system leads to a more compact and less complex system requirements specification. In [IEC61508] temporal logic is characterised as suitable for direct expression of safety requirements and as basis for formal demonstration that the expressed properties are preserved in the different development steps (compare section 5). The consequence is that already in the system requirements specification the system has to be described precisely in the form of a system model and safety requirements have to be described as part of the system requirements specification.

According to the standards a consistence check is not recommended for all cases of scenarios as well as not for all kinds of requirements, a correctness check should be done. The expenditure to perform correctness and consistence checks is not reasonable in every case when compared with its benefit. Only for high safety related functional requirements and for high safety critical operational scenarios those methods are useful.

According to standards it can be concluded that risk analysis and evidence of safe functionality are two main activities in developing system requirements specifications (see Figure 1). For both activities a precise system model is the basis. The development of this model is the third activity. These three activities together form the “building” to obtain evidence of safety in the development of system requirements specifications.

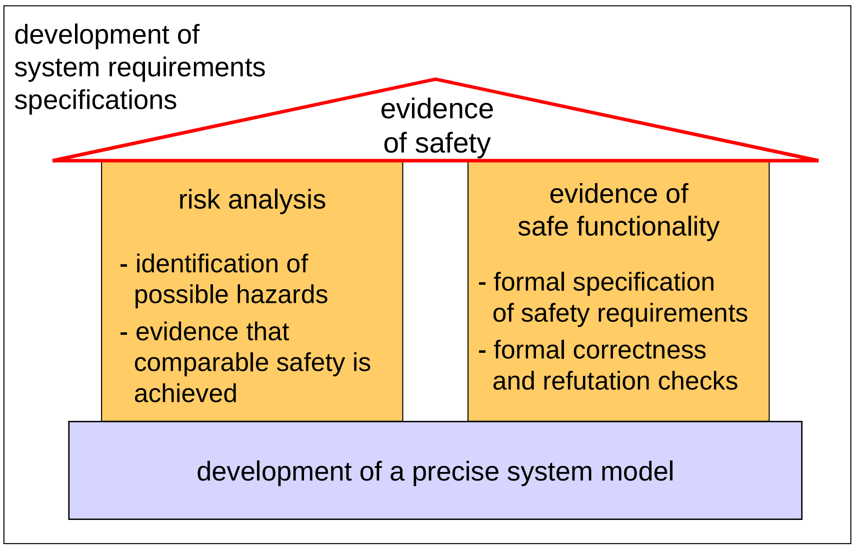

development of
system requirements
specifications
evidence
of safety
risk analysis
- identification of possible hazards
- evidence that comparable safety is achieved
evidence of
safe functionality
- formal specification of safety requirements
- formal correctness and refutation checks
development of a precise system model

Figure 1: Main activities in the development of system requirements specifications

So we can take the note that the development process of system requirements specifications should
- enable to develop a precise system model,
- support to perform risk analysis and should
- enable to obtain evidence of safe functionality:
    • by the identification of safety requirements,

• by the formal specification of safety requirements and
• by formal correctness and refutation checks.

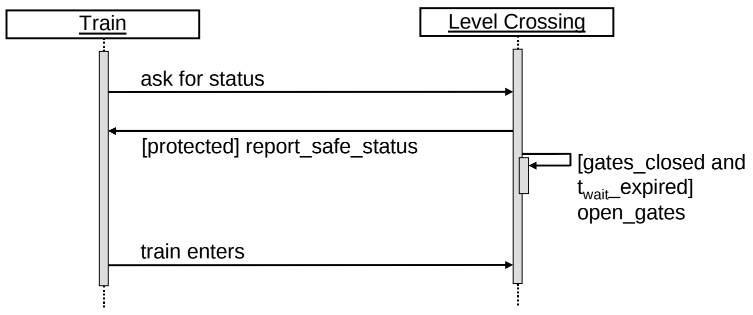
3. Development of a precise system model in UML

The use of formal methods in industrial development of safe systems is still rare. A significant barrier is that many formal languages and formal analysis techniques are unfamiliar and difficult to understand and consequently difficult to apply for engineers. But understandability plays an important role. First for developers in railway practice, who in general are no experts in formal methods, the used notations and techniques must be easy to understand, especially in teamwork. Second the assessors and the supervising authority must be able to understand the whole system model and the used techniques like correctness checks to achieve safety. That means that they have to accept and trust in the used techniques. E.g. they must be able to reconstruct correctness checks. Otherwise they would not be able to accept system requirements specification. A further important demand of the developers is usability of the methods. If intensive training is necessary to introduce methods, the acceptance of new methods is hard and leads to doubts concerning the benefit.

For these reasons existing methods must be considered, which are familiar to the developers or which become more and more important on basis of other benefits. The benefit of a new notation or technique has to be obvious. Especially the modelling notation has to be understandable and the model structure easily recognisable. This leads to clearness so that especially in teamwork the system model is easy to understand for different persons. From the engineering point of view the model used for system model should be reusable for the design. That means that the acceptance of a formal notation is low if it is different to the design notation. Nowadays in engineering graphical notations are more and more common in analysis and design stages because of its compactness and its intuitive understandability. Progressively the Unified Modelling Language (UML) becomes a standard in system and software modelling in engineering.

In [AG00] it is shown that the system model of a system requirements specification in railway practice ideally contains a description of operational scenarios, system structure and behaviour of subsystems. For execution of risk analysis ideally there is a model of the system structure and executable models. This would support the qualitative analysis. The executable model gives information concerning direct failure effects and, based on the structure, model failure interrelationships concerning system environment, system parts and subsystems can be determined.

These different aspects of system modelling can be achieved by using the following UML notations: Operational scenarios can be specified by means of sequence diagrams of UML (see Figure 2). For modelling the system structure and interfaces between system objects class diagrams are suitable (see Figure 3a). For the description of the behaviour of the system objects statecharts are used (see Figure 3b).



Train
Level Crossing
ask for status
[protected] report_safe_status
[gates_closed and t_{wait}_expired] open_gates
train enters

Figure 2: Example of an UML sequence diagram for a possible failure scenario for interaction between train and level crossing

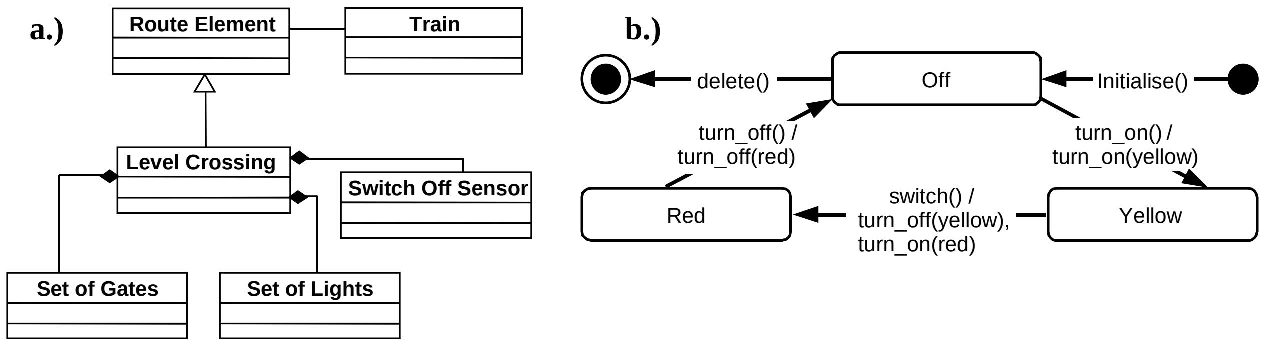


Figure 3: Example of an UML class diagram detail for a level crossing system on the left hand side (a.) and on the right hand sight an UML statechart for set of lights of the level crossing (b.)

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. For that purpose first lacks of clarity in semantics have to be identified. Second a semantic foundation has to be defined unambiguously for these lacks. This is possible by definitions of translation rules for the conversion of UML notation in a formal language, like it is used as input modelling language for model checkers. Model checking is a technique to perform correctness and refutation checks. In section 5 this is explained in more detail. In the scope of the DFG project *SafeRail* it has been developed a theory (see [Ca99]) and tool prototypes for consistence and refutation checks between UML sequence diagrams and statecharts as well as between UML statecharts and functional safety requirements, specified in temporal logic. Now our approach is adapted to the requirements-level semantics for UML statecharts of [EsWi99], which consider all special features of UML. Until now the UML tools Rhapsody of Ilogix and StP/UML (Software through Pictures/UML, see [HUSe00]) of Aonix are used in *SafeRail*. The tool StP/UML contains an easily controllable code generation. By templates the code generation is controlled concerning the code constructs to be generated. This way StP/UML is flexible related to the generated language and on how it shall be coded in detail. This kind of code generation we use to transform UML class diagrams and statecharts into the input modelling language of a model checker. The transformation rules are specified in the code generation templates. For model checking the model checkers µcke and SMV has been used. For consistence checks the statecharts model first is complete converted in a transition system (see [Ca99]). By this transformation the formal semantic is determined. That is also the basis for the translation into the input modelling language of the model checker. The semantics of sequence diagrams is determined by conversion into sequences in µ-calculus. The representation in µ-calculus is the precondition for processing by the model checker µcke. In addition the determined unambiguous semantic foundation of the graphical notations must be defined understandably and clearly for engineers so that it can be used with the correct intention. This can be reached, by using explanations in natural language. We have carried out this for example for precise definition of relations in UML class diagrams (association, composition, aggregation, heredity), see [Ar02].

4. Risk Analysis

According to [EN50129] risk analysis essentially consists of four steps:

Step 1:	system definition
Step 2:	identification of operational hazards
Step 3:	consequence analysis
Step 4:	risk assessment

In the step of risk analysis of the process model redundant activities related to other steps have to be avoided. For example it is ineffective to specify the system in step 1 again, when it is already modelled in UML notation. The more precise the system definition is the more exact are the other steps of risk analysis. In this section an approach is introduced in which risk analysis is supported by notation of the system model.

In section 3 it has already been explained that system definition contains descriptions of operational scenarios, the system structure and the behaviour of subsystems. In this section it is shown, that this system modelling aspects are ideal for carrying out risk analysis. On that basis the identification of operational hazards (step 2) can be determined systematically by using FMEA (Failure Mode and Effect Analysis, see [IEC60812]), FTA (Fault Tree Analysis, see [IEC61025]) and by analysis of scenarios of the system operation. At FMEA the consequences of failures of subsystems or failures of system functions are considered. These could lead to operational hazards, which are determined by sequence effects of such failures. On basis of regulations, standards and expert knowledge in railway area it can be decided, which top-level system failures derived by using FMEA may be relevant as operational hazards. The result of the FMEA can be checked by using FTA. The consideration of scenarios, in which failures occur, helps in determination of hazardous situations. This way it can be analysed in which cases operational hazards occur. Those scenarios can be performed with help of UML sequence diagrams.

The first step of the FMEA is a description of the system structure. This is done, by dividing the system in subsystems. The second step is the assignment of system functions to the several determined subsystems. On that basis possible failure functions can be determined to every system function. These failure functions are the basis for the data for the analysis of failure consequences and failure correlation.

These analyses can be supported by the system definition in form of the UML model. Systems and subsystems of FMEA can be derived from the objects of the UML models. The system structure can be determined on basis of the UML class diagram (compare Figure 3a and Figure 4) and the system functions are derived from class operations and actions of statecharts.

Failure consequences and correlation can be indirectly gained by the information of the UML model concerning the correlation of system functions. In UML statecharts and in these sequence diagrams, which describe interactions between system objects, the causal and temporal system behaviour can be found. This gives information concerning possible failure correlation. Especially the consideration of scenarios, in which failures occur, helps in the determination of hazardous situations. So the main creative part of the FMEA is the determination of possible failures of subsystems and of system functions and the analysis of possible failure consequences.

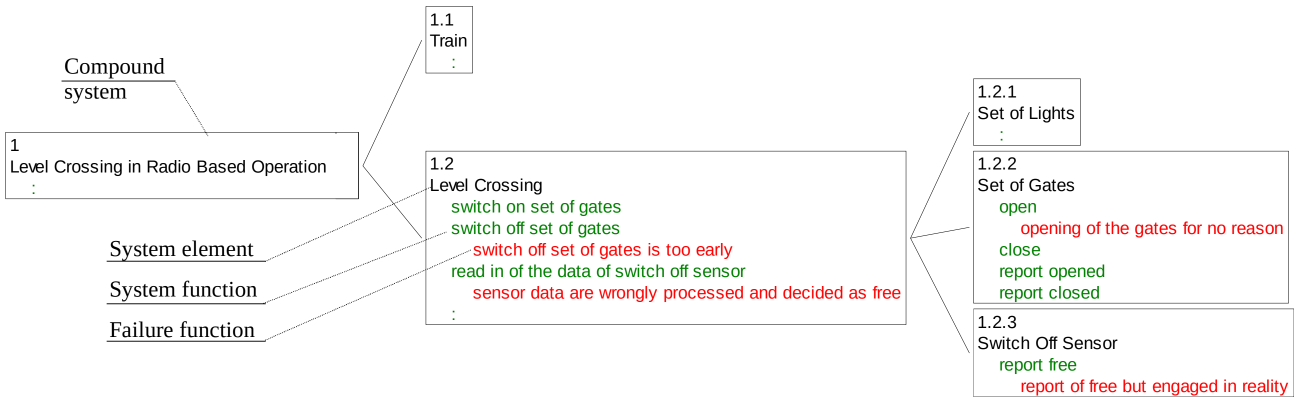


Figure 4: FMEA details corresponding with UML class diagram of Figure 3a

The result of the FMEA can be checked by analysis of the opposite way by using FTA. That means, proceeding from operational hazards, failure causes of failures of sub systems or failures of single actions, which lead to these operational hazards are identified (see Figure 4). In this way, additional failure correlation could be determined. In [prEN50129] it is recommend emphatically that two systematic methods have to be applied to be sufficiently certain concerning the completeness of the identification of hazards. In the FTA AND and OR operations of the failure causes are considered. This leads to failures on subsystem level, which are hazards on subsystem level. At FTA it is distinguished between

- *failure events* (graphical symbol: rectangle), which are caused by other events,
- *elementary events* (graphical symbol: circle), which have no causes or whose causes are not of interest, e.g. in risk analysis hazards on subsystem level, and
- events, whose causes are still not clarified (graphical symbol: rhomb).

At FMEA, as described, to every system function possible failure functions are determined. These failure functions are the basis for the data for the analysis of failure causes and failure correlation. These data also can be used to perform an FTA in an effective way.

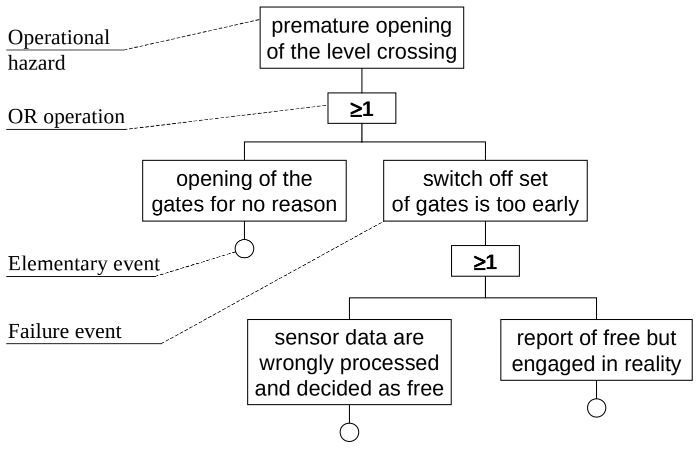

Operational hazard
premature opening of the level crossing
OR operation
≥1
opening of the gates for no reason
switch off set of gates is too early
Elementary event
≥1
Failure event
sensor data are wrongly processed and decided as free
report of free but engaged in reality

Figure 5: Example of an FTA detail for a level crossing system

A third technique to identify operational hazard is to describe operational scenarios in sequence diagram of UML and to analyse if failures in such operational scenarios lead to operational hazards. This way UML notation supports directly the second step of risk analysis. For example the sequence diagram of Figure 2 leads to the operational hazard that the train enters the level crossing even though the gates are being opened or even though the gates are open. So the operational hazard is that *the train enters an unprotected level crossing.*

From this basis functional safety requirements can be derived. To prevent systematic failures of system functions, which lead to operational hazards, functional safety requirements are formulated, which must be fulfilled by the UML system model. If necessary the UML system model must be changed. For the explained example of operational hazard the corresponding safety requirement is:

The train has to begin retardation if it has not reached the level crossing after the time ($t_{wait} - t_{braking\_distance}$) *it had received the report of safe status from level crossing.*

t_{wait} is the time, which has to be waited before the opening of the gates for the arrival of the train. The train needs the time $t_{braking\_distance}$ for braking distance. If the train does not arrive the gates will have to be opened because people do not have unlimited patience to wait in front of the level crossing. If the gates are not opened after a certain time they would enter the level crossing.

If in this way functional safety requirements are derived, the advantage is that they are usable for correctness proofs. Correctness proofs will only be possible if the safety requirements are formulated in context to the operational system model, which shall be verified. The functional safety requirements are formulated in this way, because the failure functions and hazards are identified on basis of the system model in UML notations. For that reason requirements derived this way are model specific safety requirements. The deriving of safety requirements by use of the FMEA and FTA technique is explained in more detail in [BtCaMk00].

It is decided with help of regulations, standards and expert knowledge in railway area, which determined top-level failures may be relevant as operational hazards. By use of consequence analysis (step 3 of risk analysis) and risk assessment (step 4) it is decided if the determined possible hazards are tolerable. By use of consequence analysis (step 3) the consequences of the operational hazards are determined. This way the possibilities of damages or losses, which arise from the several operational hazards, can be estimated. So it can be determined for which causes suitable functional safety requirements must be specified to avoid hazards, which are considered as intolerable or undesirable but relevant for unsafe situations.

In the project *SafeRail* the consequence analysis is realised by using ETA (Event Tree Analysis, see [DIN25419]). At ETA the analysis is based on events and conditions, which may influence the system and possible consequences are examined (see Figure 6). The event tree is developed from left to right or from bottom to top. For the different possible events the consequence events are analysed for the case that the possible event occurs and for the case that it does not occur. Therefore different possible paths are caused. On that basis possible accident sequences can be determined.

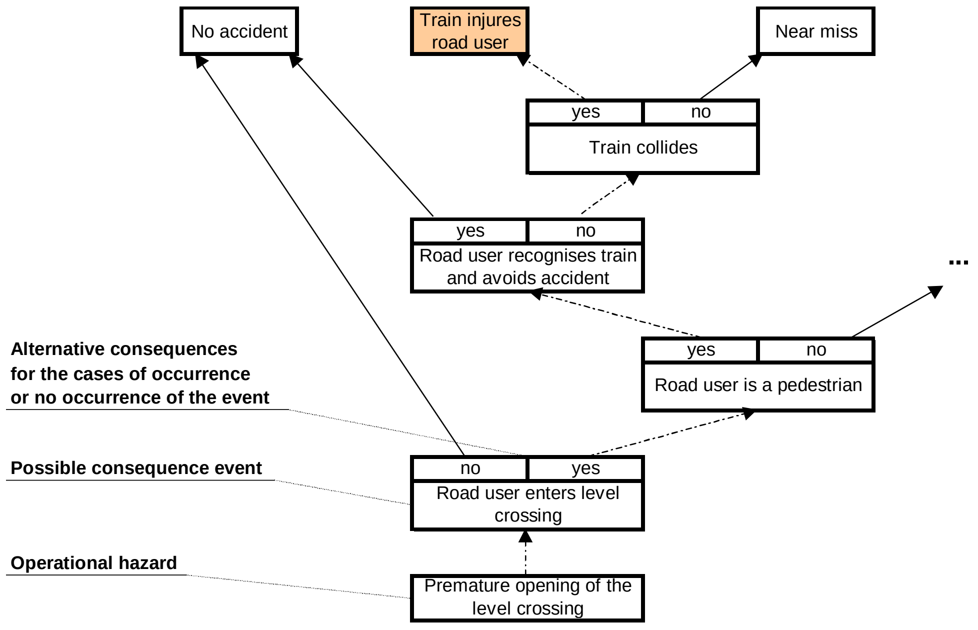

No accident
Train injures road user
Near miss
yes
no
Train collides
yes
no
Road user recognises train and avoids accident
...
Alternative consequences for the cases of occurrence or no occurrence of the event
yes
no
Road user is a pedestrian
Possible consequence event
no
yes
Road user enters level crossing
Operational hazard
Premature opening of the level crossing

Figure 6: Example of an ETA detail for a level crossing system

In the consequence analysis, those events that are relevant for interactions between the system and its environment are considered. Therefore the ETA can be supported by the UML system model this way that such events are identified, which play a part in the interaction of the system with its environment. Those events may cause possible consequences of operational hazards. On the other side this consideration of possible event sequences leads to events which had not been considered yet in the UML model and which would lead to completions to the model. In addition to this the performance of the consequence analysis can be supported by such use cases in UML notation, which describe the interaction of the system with its environment. Those use cases, which contain a consequence of an operational hazard, describe one path of the event tree in ETA. In this way UML notations may support the development of event trees. In the ETA of Figure 6 the path with broken lines corresponds with the UML sequence diagram of Figure 2.

At risk assessment (step 4) the risk of such operational hazards is estimated on the basis of the consequences, which lead to an accident, described in the consequence analysis. By using quantitative risk analysis the estimation can be performed, based on probabilistic values more precisely. In [Br01] and [Si01] the performance of a quantitative risk analysis in railway area on the basis of accident statistics is explained. By determination of operational hazards on that basis the result of risk analysis is first that the safety target for the system is identified, which must be considered at the design and the realisation of the system. Second, functional safety requirements that are specific for the system model and suitable for correctness or refutation proofs, are determined. Third safety relevant scenarios are identified, which must be consistent to the system behaviour, modelled in UML class diagrams and statecharts.

5. Evidence of safe functionality

For evidence of safe functionality correctness, refutation and consistence checks can be performed by using model checking (see [Ca99] and [Bt01]). It enables the check of the fulfilment of functional safety requirements and scenarios in operational models. At model checking, algorithms pass through the state space of the operational model, checking the compliance of the requirements or required sequences. The advantage is that it is easily applicable as method for correctness check because the check is performed automatically. But in some cases the state space of the operational model explodes and cannot be checked completely. In these cases the operational model has to be reduced otherwise model checking could only be used as refutation method to detect faults (see [Ru00]).

A precondition for formal correctness and refutation checks for safe functionality is that safety requirements have to be specified in a formal logic language (a kind of temporal logic). One reason for the very rare use of such kind of evidence of safety is that understanding and applying temporal logics is difficult for engineers, who normally are no experts in higher logic. To handle this problem there are some approaches using graphical notations. The meaning of the logical statement is graphically illustrated by the specification. Nevertheless, these approaches require dealing independently with temporal logic semantics, although in restricted form.

These difficulties are met by using pre-specified generic safety requirements (see [Bt01]). These patterns contain only those temporal logic expressions, which are suitable for the different kinds of safety requirements. Because these patterns are used for specification of safety requirements they are called safety patterns. Those safety patterns are listed in a catalogue of safety patterns. In a first step the developer identifies the suitable safety pattern with the respective formal formula in this catalogue. The identification is supported by an arrangement of the safety patterns based on a semantic classification of different kinds of safety requirements. In a second step the developer has to adapt the identified safety pattern to the respective safety requirement in context to the operational model. So the result is a formally specified safety requirement, which is an instance of a safety pattern.

In the catalogue every safety pattern is given in the formal notations CTL (Computation Tree Logic) and LTL (Linear Time Temporal Logic). Furthermore the specification in μ-calculus and Temporal OCL (an extension of the Object Constraint Language by temporal logic aspects, compare [Fl01]) is in process. This way the developer is able to choose the formalism required for the model checker according to his preferences. For that reason different expressive powers of different variants of temporal logic are considered in the safety patterns approach. Every safety pattern is explained in natural language, so that the meaning of the safety patterns is easily to understand.

In case of formal specification an additional specification in natural language is necessarily required in the standards (see [IEC61508] and [EN50128]) for reasons of clear understandability. But the problem is that natural language permits ambiguous formulations. Therefore in natural language a requirement might be specified, which is not equivalent to the original intention of the formal specification. For that reason safety patterns do not only support formal specification of safety requirements but also enable specifications in words of natural language equivalent to the formal specification. Every safety pattern contains a pattern in a restricted natural language. The meaning of the used constructs and words, which are used for description, are fixed in the catalogue (see [BtGo02]). That means that a norm language is used for the patterns in natural language and the descriptions in natural language. A norm language sounds like a natural language, but it is a strongly reduced form of natural language. It is a connecting link between natural languages and formal languages (see [Or97]). This way a safety pattern can also be used to formulate precisely a safety requirement in natural language. Especially in teamwork, clearly understandable and precisely formulated specifications of safety requirements are a precondition for the development of safe automation systems. Besides in communication with approval authorities the meaning of formal safety requirements is easy to understand based on such a safety pattern catalogue. Consequently a main advantage in comparison to other approaches is that safety patterns imparts expert knowledge with regard to formal specification, formulation of safety requirements in a restricted form of natural language and with regard to temporal logic characteristics of the different kinds of safety requirements. In Figure 7 an example of a safety pattern is shown.

Safety Pattern ds-wet-27

dynamic safety requirement without explicit time, concerning beginning of validity and concerning permissibility of validity, safety pattern no. 27

Classification

dynamic safety requirement / safety requirement without explicit time / safety requirement with temporal dependencies / safety requirement concerning beginning of validity / safety requirement concerning permissibility of validity / validity strict after a certain point in time / validity prohibition before point in time / validity any times often

Pattern in Natural Language

Only strictly after that point in time when *p* occurs then it is permitted that *q* is valid any times often.

Alternatives:

Only strictly after the event *p* has occurred then from now on it is permitted that the action *q* is being executed.

If the event *p* has not occurred until now, then the action *q* must not yet be executed.

Formal Pattern

CTL: `A((not q) W (p and (not q)))`

LTL: `(not q) W (p and (not q))`

Temporal OCL[3]: `inv: let requiredSequence = Sequence{not qConf, pConf} in begin implies self@post→forAll(s:OclPath | s→includes(requiredSequence) or s→excludes(qConf))`

Explanation in Natural Language

A main characteristic of this safety pattern is that *q* may only occur strictly after *p*. The exact point in time after *p* is irrelevant. The main thing is that *q* occurs strictly after *p*. That means *q* must neither occur before nor together with *p*. *q* may occur but it do not have to.

Example of Use

System

Level crossing

Safety requirement in natural language

The gates may only be opened when the train has passed the level crossing.

Safety requirement in norm language

Only strictly after the event *train.train_passed* has occurred then from now on it is permitted that the action *level_crossing.opening* is being executed.

Safety requirement in formal language

CTL: `A((not level_crossing.opening) W (train.train_passed and (not level_crossing.opening)))`

Figure 7: Example of a safety pattern

[3] "*p*Conf" describes the configuration, which is reached at the occurrence of the event *p* and "*q*Conf" is the configuration, which is valid at the execution of the action *q*. Configurations describe unambiguously every possible system state and in this way they also represent certain events and actions.

It is also in process to enlarge the data in every safety pattern with graphical descriptions. The graphical description contains typical possible sequences of states and, depending on the respective formal language also different examples of possible computation paths, see [BtGo02]. To support the identification of the suitable safety requirements a tool-supported dialog guidance is in development, compare [BtGo02]. With the help of this tool the suitable safety pattern can be identified through several dialogs and through different static and dynamic dialog forms. In the dialogs different classification trees of safety patterns are considered. Such a tool will also support the differentiation of similar safety patterns.

Altogether we can take the note of the following benefits of safety patterns. Safety patterns support

- to specify correctly safety requirements in formal languages.
- to interpret correctly formal specifications.
- the use of different tools for correctness checking.
- to specify safety requirements by using a restricted terminology of natural language corresponding to the formal specification.

6. Conclusion

In this paper the main activities at the development of system requirements specifications have been derived from demands of new European standards in railway area and from the international generic standard [IEC61508] concerning safety critical systems. Not only a safe system requirements specification has to be achieved but also it has to be demonstrated that a safe system specification is achieved in system requirements specification according to these standards. To obtain evidence of safety the importance of risk analysis and the importance of evidence of safe functionality according to standards have been shown. A precise system model is the basis for both activities.

In conclusion the overview on the process model for the development of system requirements specifications for railway systems is shown in Figure 8. The arrows show the flow of information between activities.

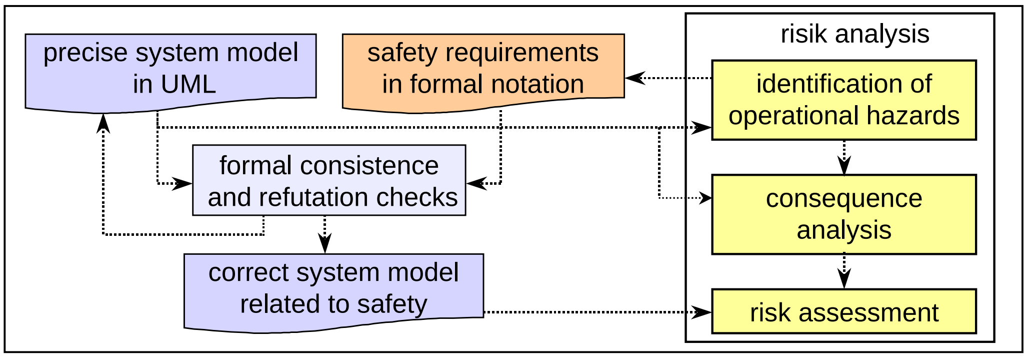

precise system model in UML
safety requirements in formal notation
risik analysis
identification of operational hazards
formal consistence and refutation checks
consequence analysis
correct system model related to safety
risk assessment

Figure 8: Overview on the process model for the development of system requirements specifications for railway systems

The UML notation can be used to specify a precise and clearly understandable system model, which is exact, unambiguous and compact as well as easily understandable by developers, operation authority, supervising authority and assessors. On the one hand this is the basis to support risk analysis in an efficient way. The several steps of risk analysis necessary for preparation of a system requirements specification have been explained. The meaning of the safety analysis techniques ETA, FTA and FMEA as part of risk analysis has been introduced. On the other hand a precise system model in UML enables correctness and refutation checks related to the fulfilment of safety requirements. By risk analysis the identification of safety requirements can be supported. Using safety patterns, engineers who are no experts in higher logic, are able to specify and to interpret correctly safety requirements in formal specification languages, what is recommended by standards, as well as precise specifications in natural language. In this way safety patterns are a connecting link between specifications in natural language and formal specifications. Within the scope of the DFG project *SafeRail* a theory (see [Ca99] and [BtGö02]) and tool prototypes have been developed for consistence and refutation checks between UML statecharts and functional safety requirements specified by using safety patterns. In case of detected failures the system model has to be changed. A correct system model related to safety is the basis for risk assessment (see [Br01]). Further on there will be developed conversions, which provide the full expressive power of UML statecharts. In addition information of UML class diagrams shall be considered according to semantics described in [Ar02].

7. References

[Ar02] Arabestani, S.: *Relations in Object-Oriented Analysis*. In H. Ehrig and M. Große-Rhode (eds.): INT 2002, 2nd International Workshop on Integration of Specification Techniques for Applications in Engineering, Grenoble, 2002, pp. 37-47.

[Br01] Braband, J.: *Lessons Learned from Risk Assessment Studies in Railway Signalling.* In J. Pachl (Hrsg.): Moderne Sicherheitsanalysen - Theorie und Praxis, IfEV Schriftenreihe, TU Braunschweig, Heft 64, 2001, S. 3-16.

[BtCaMk00] Bitsch, F.; Canver, E.; Moik, A.: *Strukturierte Erstellung von Sicherheitsspezifikationen in UML mit Hilfe der FMEA-Methode*. In E. Schnieder (Hrsg.): Forms '99 - Formale Techniken für die Eisenbahnsicherung, Fortschritt-Berichte VDI, Reihe 12, Verkehrstechnik/Fahrzeugtechnik, Nr.436, VDI Verlag GmbH, Düsseldorf 2000, S. 225-245.

[Bt01] Bitsch, F.: *Safety Patterns - The Key to Formal Specification of Safety Requirements*. In U. Voges (ed.): Proceedings of 20th International Conference, SAFECOMP 2001 - Computer Safety Reliability and Security, Budapest, September 2001, LNCS 2187, Springer-Verlag, Berlin Heidelberg, 2001, S. 176-189.

[BtGo02] Bitsch, F.; Göhner, P.: *Spezifikation von Sicherheitsanforderungen mit Safety-Patterns.* Software Engineering in der industriellen Praxis, VDI-Bericht-Nr. 1666, Düsseldorf, 2002, S. 29-40.

[Ca99] Canver, E.: *Einsatz von Model-Checking zur Analyse von MSCs über Statecharts*. Ulmer Informatik Berichte 99-04, Universität Ulm, 1999.

[DIN25419] Normenausschuss Kerntechnik: *Ereignisablaufanalyse: Verfahren, graphische Symbole und Auswertung*. 1985.

[EN50126] CELENEC EN 50126: *Railway Applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)*. 1999.

[EN50128] CELENEC EN 50128: *Railway Applications - Communications, signaling and processing systems - Software for railway control and protection systems*. 2001.

[prEN50129] CELENEC prEN 50129: *Railway Applications - Safety related electronic systems for signaling*. 2000.

[EsWi00] Eshuis, R.; Wieringa, R.: *Requirements-level semantics for UML statecharts*. In Proc. FMOODS 2000, IFIP TC6/WG6.1. Kluwer, 2000.

[Fl01] Flake, S.; Mueller, W.: *An OCL Extension for Real-Time Constraints*. In T. Clark and J. Warmer (eds.): Advances in Object Modelling with the OCL, Springer-Verlag, Heidelberg, 2001.

[HuSe00] Huber, S.; Seidl, H.G.: *Architecture Centric Development - generating more code from your UML models*. Aonix Europe Ltd., White Paper, 2000.

[IEC61025] International Standard IEC 61025: *Fault Tree Analysis (FTA).* International Electrotechnical Commission, Geneva, Switzerland, 1990.

[IEC60812] International Standard DIN IEC 60812: *Analysis Techniques for System Reliablity - Procedure for Failure Mode and Effects Analysis (FMEA).* International Electrotechnical Commission, Geneva, Switzerland, 1985.

[IEC61508] International Standard IEC 61508: *Functional Safety of Electrical/Electronic/ Programmable Electronic Safety Related Systems*. International Electrotechnical Commission, Geneva, Switzerland, 2000.

[KaLeSu00] Kammel, K.; Lennartz, K.; Suwe, K.-H.: *Die Anwendung von formalen Techniken aus Sicht des Eisenbahn-Bundesamtes.* In E. Schnieder (Hrsg.): Forms '99 - Formale Techniken für die Eisenbahnsicherung, Fortschritt-Berichte VDI, Reihe 12, Verkehrstechnik /Fahrzeugtechnik, Nr.436, VDI Verlag GmbH, Düsseldorf 2000, S. 55-69.

[No01] Nordland, O.: *Presenting a Safety Case – A Case Study*. In U. Voges (ed.): Proceedings of 20th International Conference, SAFECOMP 2001 - Computer Safety Reliability and Security, Budapest, September 2001, LNCS 2187, Springer-Verlag, Berlin Heidelberg, 2001, pp. 56-65.

[Or97] Ortner, E.: *Methodenneutraler Fachentwurf - Zu den Grundlagen einer anwendungsorientierten Informatik.* B.G. Teubner Verlagsgesellschaft, Stuttgart, Leipzig, 1997.

[Ru00] Rushby, J.: *Disappearing Formal Methods*. Invited paper from the Fifth IEEE International Symposium on High Assurance Systems Engineering (HASE), pp. 95-96, November, 2000, Albuquerque NM.

[Si01] Six, J.: *Ableitung von Zahlenwerten für Risikoanalysen aus der Unfallstatistik.* In J. Pachl (Hrsg.): Moderne Sicherheitsanalysen - Theorie und Praxis, IfEV Schriftenreihe, TU Braunschweig, Heft 64, 2001, S. 17-32.