Quantitative Model-Based Safety Analysis: A Case Study

Matthias Güdemann*, Frank Ortmeier
matthias.guedemann@ovgu.de, frank.ortmeier@ovgu.de

Abstract:
The rising complexity of many safety-critical systems necessitates new analysis methods. Model-based safety analysis approaches aim at finding critical failure combinations by analysis of models of the whole system (i.e. software, hardware, and failure modes). The big advantage of these methods compared to traditional approaches is that the results are of very high significance.
Until now, model-based approaches have only to a limited extent been applied to answer quantitative questions in safety analysis. Model-based approaches in this context are often limited to analysis of specific failure propagation models. They do not include system dynamics and behavior. A consequence is, that the methods are very error-prone because of wrong assumptions.
New achievements in the domain of (probabilistic) model-checking now allow for overcoming this problem. This paper illustrates how such an approach for quantitative model-based safety analysis is used to model and analyze a real-world case study from the railway domain.

1 Introduction

Due to the rising complexity and basically ubiquitous application, more and more software intensive systems become safety-critical. The amount of software in such systems is increasing at the same time, posing an additional challenge to build these dependable and fault-tolerant.
In order to prevent as many accidents as possible, a lot of effort is being put into safety in many domains. Requirements for the development and life cycle of safety-critical systems are now specified in many different norms like the general IEC 61508 [Lad08], DO178-B [RTC92] for aviation or ISO 26262 [ISO09] for automotive. In addition to requirements, they present guidelines to make systems more fault-tolerant. All these norms require some sort of safety assessment before a system is put into operation. Many of these methods have a long history (some dating back to the 60ies). However, they are more or less only methodologies and purely rely on skill and expertise of the safety engineer. A potential source of safety-critical problems can only be anticipated if an engineer thinks of it a design time. But this foreseeing becomes ever harder, because of rising hardware and software complexity.

*Acknowledgement: Matthias Güdemann is funded by the German Ministry of Education and Science (BMBF) within the ViERforES project (no. 01IM08003C)

To counter these problems, new model-based safety analysis methods get applied. This means that a model of the system under consideration as well as its environment is built. Then the analysis is not solely based on the engineer's skill but also on the formal analysis of this model. Some examples of such methods are explained in [ABB+06, ORS06, ADS+04, BV03]. In some cases it is even possible to semi-automatically deduce cause-consequence relationships between component failures and loss of system.
These are mostly qualitative methods, which can only show fault tolerance by giving critical combinations of failure modes. But for the assessment of the dependability the most important question is: *"What is/are the probabilities of any of the systems hazards?"* For a satisfying answer, accurate quantitative safety analysis methods must be applied.
In this paper we show the application of a new, quantitative model-based safety analysis technique – probabilistic deductive cause-consequence analysis [GO10] – to a real-world case study and report on our experiences. The paper starts with a short presentation of the case study (Sect. 2). Sect. 3 explains the formal modeling and analysis of the case study and observations made from a comparison to a traditional safety analysis, Sect. 4 discusses some related work and Sect. 5 concludes the paper.

2 Case Study

The following case study of a radio-based railroad control was used as a reference case study used in the priority research program 1064 "Integrating software specifications techniques for engineering applications" of the German Research foundation (DFG), it was supplied by the German railway organization, Deutsche Bahn. The scenario addresses a novel technique for controlling railroad crossings. This technique aims at medium speed routes, i.e. routes with maximum speed of 160 km/h. An overview is given in [KT02].

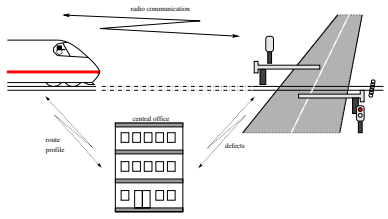


Figure 1: Radio-based railroad crossing

The main difference between this technology and the traditional control of railroad crossings is that signals and sensors on the route are replaced by radio communication and software computations in the train and railroad crossing. This offers cheaper and more flexible solutions, but also shifts safety critical functionality from hardware to software. Instead of detecting an approaching train by a fixed sensor on the track, the train continuously computes the position where it has to send a signal to secure the level crossing. To calculate this activation point the train uses data about its position, maximum deceleration and the position of the crossing. Therefore the train has to know the position of

the railroad crossing, the time needed to secure the railroad crossing, and its current speed and position. The first two items are memorized in a data store and the last two items are measured by an odometer. For safety reasons a safety margin is added to the activation distance. This allows compensating some deviations in the odometer. The system works as follows:
The train continuously computes its position. When it approaches a crossing, it broadcasts a 'secure'-request to the crossing. When the railroad crossing receives the command 'secure', it switches on the traffic lights, first the 'yellow' light, then the 'red' light, and finally closes the barriers. When they are closed, the railroad crossing is 'secured' for a certain period of time. The 'stop' signal on the train route, indicating an insecure crossing, is also substituted by computation and communication. Shortly before the train reaches the 'latest braking point' (latest point, where it is possible for the train to stop in front of the crossing), it requests the status of the railroad crossing. When the crossing is secured, it responds with a 'release' signal which indicates, that the train may pass the crossing. Otherwise the train has to brake and stop before the crossing. Behind the crossing, a sensor detects that the train has passed and an 'open' signal is sent to the crossing. The railroad crossing periodically performs self-diagnosis and automatically informs the central office about defects and problems. The central office is responsible for repair and provides route descriptions for trains.

3 Model-Based Safety Analysis

Our model-based safety analysis consists of building a formal system model which can then be analyzed using formal analysis techniques based on temporal logics and model checking. The accuracy of the formal system model is the most important factor in the accuracy of the whole analysis, especially the modeling of the probabilistic behavior.

3.1 Building a formal model

For model-based safety analysis it is necessary to model (a) the controlling software, (b) the controlled hardware, (c) the environment and (d) possible failure modes. We can not show the full case study here, but only show one part of it. Fig. 2 shows a model of the crossing in state charts notation[1].
Initially the barriers (of the crossing) are *opened*. When the crossing receives a close request from an arriving train - i.e. condition *Close* becomes true, the barriers start *closing*. This process takes some time. After a certain amount of time the barriers are *closed*. They will remain closed until the train has passed the crossing (detected by a sensor). The barriers reopen automatically after a defined time interval. This is a standard procedure in railroad organization, as car drivers tend to ignore closed barriers at a railroad crossing if

[1] In this paper, we use a basic (but very precise) semantics for state charts. Basically no queues are allowed, events do not persist and parallel composition is synchronous. This semantics is very intuitive on the one hand and very easy to translate into model checker input languages on the other hand.

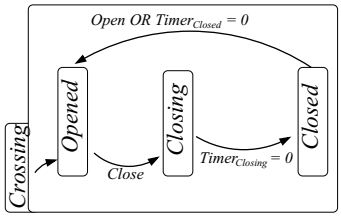


Figure 2: Model of the crossing

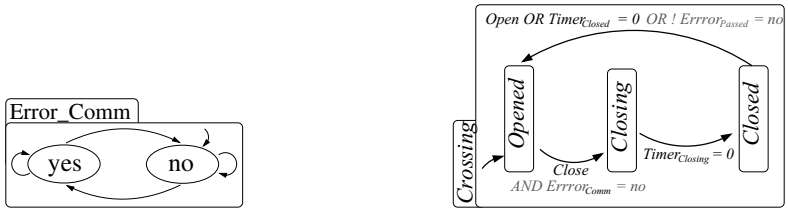
the barriers are closed too long. So it is better to reopen the barriers, than having car drivers slowly driving around the closed barriers. Analogously the other parts of the system (i.e. the train, radio communication, brakes, cars, control software) are modeled. Together this forms a functional model of the system. This means it is a model of the system in an ideal world, where no errors occur.
The safety goal of the system is clear: it must never happen that the train is on the crossing and a car is passing the crossing at the same time. A well designed control system must assure this property at least as long as no component failures occur. The corresponding hazard H is "a train passes the crossing and the crossing is not secured". This is the only hazard which we will consider in this paper.

3.2 Modeling failure modes

The next step is to extend this model such that failures are also correctly described. In this paper we do not address the question which failure modes are necessary to model[2], we assume, that this has already been determined. For the example the following six failure modes are considered: **failure of the brakes** (*error_brake*) which describes the failure of the brakes, **failure of the communication** (*error_comm*) which describes the failure of the radio communication, **failure of the barriers closed sensor** (*error_closed*) which describes that the crossing signals *closed*, although it is not closed, **failure of the barriers' actuator** (*error_actuator*) which describes that the actuator of the crossing fails, **failure of the train passed sensor** (*error_passed*) which describes that the sensor detecting trains which passed the crossing fails and **deviation in the odometer** (*error_odo*) which describes that the odometer does not give 100% precise data.
These failure modes are integrated into the formal model. Modeling of failures can always be split into two (sub-)tasks: modeling of the occurrence pattern and modeling of the direct effect of the failure mode. *Occurrence patterns* describe how and when a given failure mode may occur and are modeled by failure charts.
The most basic failure chart has two states, one state *yes* modeling the presence of the failure and one state *no* modeling its absence. The transitions between these states determine

[2]There are numerous other techniques for answering this question. They range from experienced based approaches like component specific list of failure modes to formally grounded methods like failure-sensitive specification [OR04].

the type of failure mode. For example: if the state *yes* can never be left once it became active, the failure mode is called *persistent*. If the state may non-deterministically switch between *yes* and *no*, it is called *transient*. More complex occurrence patterns (e.g. broken until repair) are also possible. Which occurrence pattern is best fitting for a given failure mode is a design/analysis decision.
Modeling of the *direct effects of the failures* of a failure mode basically comes down to adding transitions and/or states to the functional model, which are activated or in the case of additional states, reachable, if the failure occurs (resp. if the corresponding failure automaton is in state "yes").
Correct modeling of failures is a difficult and error-prone task. From a formal point of view the semantics of the original model (without failure modes) and the extended model have little to do with each other. From an intuitive point of view, one would expect some sort of trace-inclusion, i.e. the original behavior of the functional system must still be possible in the extended system model. This property can be assured, if some syntactic rules are followed during modeling of failure modes' direct effects [OGR07].
Fig. 3(b) shows the modeling of the failure effect of the failure modes *error_comm* and *error_passed* for the crossing, Fig. 3(a) shows the transient failure automaton for *error_comm*. If it is active then state *Opened* will not be left although a *Close* signal was sent, i.e. the communication failed. The failure effect of *error_passed* is modeled such that state *Closed* may be left if the corresponding failure automaton is in state *yes*, i.e. a misdetection of the sensor happened.



(a) Failure Automaton (b) Failure Effects Modeling

Figure 3: Modeling of error_comm failure mode

3.3 Modeling for Quantitative Safety Analysis

For an accurate quantitative model-based analysis, probabilistic information must be added to the extended system model. Accurate modeling of the occurrence pattern and the occurrence probability of failure modes is very important. There exist two main types of failure probabilities. The first is *per demand* failure probability, which is the probability of the system component failing its function at a given demand (comparable to *low demand mode* in IEC 61508). The second is *per time* probability, which is the *rate* of failures over a given time interval (comparable to *high demand or continuous mode* in IEC 61508).
Which type of failure probability is best fitting for a given failure mode can only be decided on a case-by-case basis. Transient sensor failures will very often be modeled as a

per time failure mode, as they are active the whole time. Other failure modes, like the activation of a mechanical device, will often be modeled as a *per demand* failure, as a clear moment of activation exists. Per demand failures often are of persistent nature[3].

3.3.1 Discrete Time Model

The underlying semantics of the formal model is a discrete time model in which the temporal behavior of the failure modes and system model must be integrated. For each time step the system performs, an equal amount of time δt passes, which is called the *temporal resolution*. For the example case study, the formal model consists of $10km$ of railroad tracks. The average speed is $115\frac{km}{h}$. The formal system model in built in a way that this translates to a temporal resolution of $\delta t = 5s$, i.e. for every system time step, $5s$ pass.

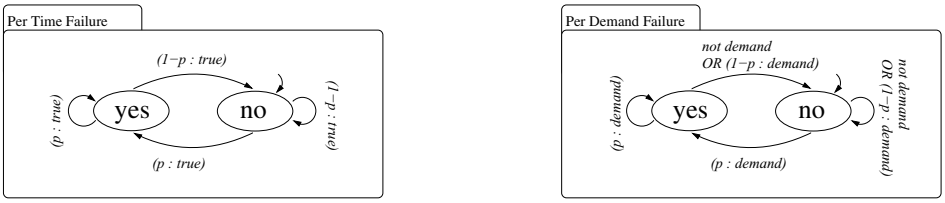
3.3.2 Modeling per time failure modes

Transitions of the failure automata are labeled with constraints of the form $(p : \phi)$ which translates to: "If ϕ holds, then the transition is taken with probability p". This can be abbreviated by omitting p which then results in probability 1. In order to be a well defined probabilistic model, for each transition label, the outgoing transition probabilities must always sum up to 1. This assures that a valid probability measure is defined.
A *per time* failure mode can be modeled by adding the failure probability to the transition from state *no* to state *yes* as shown in Fig. 4(a). Making its potentially non-deterministic behavior *probabilistic*. The activation condition is always *true* and the sum of probabilities of outgoing transitions is always 1. In the case shown, the failure automaton enters state *yes* with a probability p, stays in this state with the same probability and enters (and stays in) state *no* with probability $1 - p$.
The parameter for a per-time failure is normally given as a *failure rate* λ and interpreted as the parameter for the exponential distribution function (Eq. 1) which computes the probability that a failure occurs before time t. This continuous function can be approximated in discrete time using a per-time failure automaton as shown in Fig. 4(a). This leads to a geometric distribution function (Eq. 2) which computes the probability that a per-time failure appears in at most k time-steps. Setting time $t = k \cdot \delta t$, gives a good discrete time approximation of the exponential distribution [GO10].

$$P(X \leq t) = \int_0^t e^{-\lambda t} dt \qquad (1) \qquad\qquad P(X \leq k) = 1 - (1-p)^k \quad (2)$$

With the given δt, the failure rates for the per-time failures can be converted to transition probabilities for per-time failure automata. Modeling the $error\_passed$ as *per time* failure mode with a failure rate of $7e^{-9}\frac{1}{s}$ and the failure of the odometer with a failure rate of $6e^{-5}\frac{1}{s}$, the transition failure probabilities are $3.5e^{-8}$ for $error\_passed$ and $3e^{-4}$ for $error\_odo$. The deviation of the odometer is then modeled by choosing a deviation from the set $\{-3, -2, -1, 0, 1, 2, 3\}$. The probability of the deviation is modeled to be normally distributed with $\mu = 0\frac{m}{s}$ and $\sigma = 1\frac{m}{s}$.

[3] But there exist of course other failure modes, which are transient and should be modeled as per demand probabilities resp. persistent failure modes, which should be modeled with per time failure rates.



(a) Per-time (b) Per-demand

Figure 4: Failure Automata for quantitative failure mode modeling

3.3.3 Modeling per-demand failure modes

Correct modeling of a *per demand* failure mode is more complex. The challenge is to specify a correct occurrence pattern such that assures that the failure mode can *only* appear if a demand to the component is given and is then taken only with the given failure probability. To accomplish this, the failure probability is first added to the transitions to state *yes* and the state *no* may only be left if there is a *demand* to the component. In other words, a predicate *demand* is a necessary condition for the occurrence of the failure mode. A failure automaton for a *per demand* failure mode is shown in Fig. 4(b).
The failure mode *error_comm* is integrated in this way. In this case, there is a demand if the crossing is in state *Opened* **and** the *Close* signal is sent. If *error_comm* is not present, then the crossing enters state *Closing*, if the failure mode is present or no *Close* signal is sent, the state *Opened* is not left. When *demand* holds $((Crossing = Opened) \wedge Close)$, the failure automaton can change to state *yes* with probability p or stay in state *no* with probability $1 - p$. This means that in the next time step, it is possible that the crossing is either in *Opened* or *Closing*. To model this correctly, the state *Undecided* is introduced which represent either of these states.
In addition a *decide* automaton as shown in Fig. 5(b) is added, which changes from state *undef* to *[Opn, Clg]* if *demand* holds. Based on these two automata the predicates $in(Opened)$ and $in(Closing)$ are defined as shown in equation (3) and (4) which indicate the "real" state of the crossing, *Crossing'* refers to the model of the crossing with per-demand failure mode modeling.

$$\begin{aligned} in(Opened) \quad &:= \quad Crossing' = Opened \vee (Crossing' = undecided \\ &\quad \wedge decide = [Opn, Clg] \wedge \neg failure = no) \qquad (3) \\ in(Closing) \quad &:= \quad Crossing' = Opened \vee (Crossing' = undecided \\ &\quad \wedge decide = [Opn, Clg] \wedge failure = no) \qquad (4) \end{aligned}$$

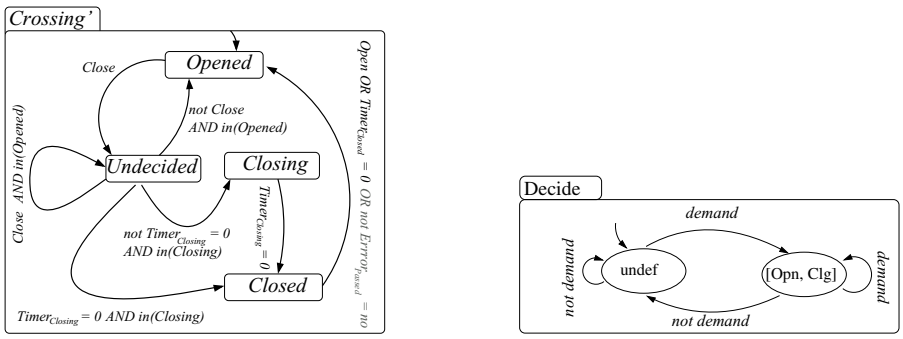
As *Undecided* represents either state *Opened* or *Closing*, depending on the state of the failure automaton, the transitions of these states must be added to the state *Undecided* , in conjunction with the the above predicates.

- For the state *Opened*

- to state *Opened*, activation condition $\neg Close \wedge in(Opened)$, i.e. in state *Opened* and there is no demand
- to state *Undecided*, activation condition $Close \wedge in(Opened)$, i.e. there is a demand that possibly fails

- For the state *Closing* the following transitions must be added to *Undecided*:
  - to state *Closing*, activation condition $\neg Timer = 0 \wedge in(Closing)$
  - to state *Closed*, activation condition $Timer = 0 \wedge in(Closing)$

The complete modeling of the *per demand* failure effect for *error_passed* is shown in Fig. 5(a). The decide automaton, *Crossing* and the per-demand failure automaton together show the same *observable* behavior as the qualitative failure modeling. It becomes clear why such a complex construction is necessary: the state *Opened* has successor states if there is a *demand* and successors states if there is no *demand*. Therefore only being in state *Opened* is not sufficient to specify the *demand* predicate, because the failure automaton could then make a transition at the wrong time, resulting in wrong overall probabilities, especially important for persistent failures which cannot disappear after their occurrence. We give an example trace that shows the relevant predicates and states of the automata to illustrate that the resulting traces of the probabilistically modeled system are the same as for the system for qualitative analysis, if the *per demand* failure modes are correctly integrated as described. The first trace shows the behavior of the extended system model



(a) Failure Effects Modeling (*Crossing'*)

(b) Decide Automaton

Figure 5: Modeling of per-demand failure mode error_com

without explicit modeling of the *per demand* failure mode. The failure appears after the first time step, but has no effect there because there no *Close* command was sent. At the third time step, the failure appears again, now it has an effect (*Close* command was sent) causing the crossing to stay in state *Opened*. Afterwards the failure disappears, *Close* is sent again and the crossing enters state *Closing*. The following trace is the corresponding trace of the extended system model with probabilistic modeling of the *per demand* failure. Here the failure can only appear at the fourth time step, as there was no *demand* before. This causes the *decide* automaton to enter the state **[Opn,Clg]** and the crossing to enter

time-step	1	2	3	4	5
error_comm	no	yes	yes	no	no
Crossing	*Opened*	*Opened*	*Opened*	*Opened*	*Closing*
Close	$false$	$false$	$true$	$true$	$false$

state *Undecided*. The failure is active at the fourth time step, therefore *in(Opened)* holds and the "real" state of the crossing is *Opened*. At the fifth time step, the *Close* command is sent again, this time the failure disappears and *in(Closing)* holds. Both traces show the

time-step	1	2	3	4	5
error_comm	no	no	no	**yes**	**no**
Crossing'	*Opened*	*Opened*	*Opened*	***Undecided***	***Undecided***
decide	*undef*	*undef*	*undef*	**[Opn,Clg]**	**[Opn,Clg]**
in(Opened)	$true$	$true$	$true$	**true**	$false$
in(Closing)	$false$	$false$	$false$	$false$	**true**
Close	$false$	$false$	$true$	$true$	$false$

same observable behavior, i.e. the states of the crossing and therefore the failure effects are the same in both cases. Nevertheless it can easily be seen that the failure mode can only appear at the time of a demand, whereas in the modeling without per demand failures the failure can appear at any time, but its effect manifests only at certain time steps (the demands in the probabilistic modeling). This illustrates again that in order to compute accurate probabilities, this distinction is necessary although the correct integration in not trivial. A much more detailed discussion about the integration of both per-demand and per-time failure modes can be found in [GO10].

3.4 Analysis Results

The constructed probabilistic model was analyzed using the PRISM model checker using hybrid models (sparse matrices and MTBDDs) on an Athlon 64 X2 4800+ CPU with 2 Gbyte of RAM. The overall model consisted of 11295021 reachable states and 518674960 transitions resulting from the parallel composition of the state machines of the model description. The duration of the analysis was approx. 190 minutes and 279.7 MByte of RAM were allocated. The hazard H was defined as "the position of the train is on the crossing and the crossing is not in state *closed*". Equation (5) shows the result of the quantitative analysis. The probabilistic temporal logic formula is explained in [GO10].

$$P(H) = 4.47363 \cdot 10^{-6} \tag{5}$$

Compared to a more traditional analysis method based on the a-posteriori estimation of the hazard probability on the resulting critical combinations of failures (for details see [ORS05]), this result is much more accurate. The following equation shows the result using the standard approach for quantitative fault tree analysis (FTA) [VDF$^+$02] as

shown in formula (6).

$$P_{FTA}(H) \leq \sum_{\Delta \in mcs} \prod_{\delta \in \Delta} P(\delta) \quad (6)$$

$$P_{FTA}(H) \leq P(error\_passed) + P(error\_odo) + P(\text{cut sets of size } \geq 2)$$

$$P_{FTA}(H) \leq 2.8 \cdot 10^{-6} + 2.5 \cdot 10^{-2} + O(1 \cdot 10^{-14}) \approx 2.5 \cdot 10^{-2} \quad (7)$$

For this calculations, the probabilities for $error\_passed$ and $error\_odo$ must be estimated for *one* train passing the crossing. In the example, it was assumed that the train needs roughly[4] 400s. These coarse results (Eq. (7)) could then be refined with constraint probabilities. For example, one could deduce that only deviations of at least $2\frac{m}{s}$ are dangerous. This would then lower the probability for $error\_odo$ by the order of 2. It is still rather obvious that this estimation is very coarse compared to quantitative model based analysis. Another drawback of the traditional approach is that if the system is not carefully analyzed for temporal behavior, the estimation can easily be too optimistic. Just try to decide if a deviation of $3\frac{m}{s}$ could also be justified or not.
The biggest advantage of the model-based quantitative analysis is that the stochastic dependencies[5] which often are inherent in such a system are *automatically* considered in the system model, whereas all a-posteriori methods depend on the stochastic *independence* which is often not satisfied.

4 Related Work

Some related model-based analysis methods like [BV03, ABB+06, ADS+04] have already been mentioned. Together with basic fault tree analysis [VDF+02] they have the disadvantage that the *results* of a qualitative analysis are used for quantitative estimations. As already discussed, these estimations rely on the assumption of stochastic independence, which is often not fulfilled.
Notable approaches for doing quantitative safety analysis directly on a formal system model is presented in [BPRW08]. In this work, critical combinations of failures, analogously to the minimal cut sets of FTA or minimal critical sets of DCCA are analyzed for their relative impact. They are sorted according to their contribution to the overall hazard probability. The difference is that the analyzed system behavior is limited to allow *only* failure modes in such a set. This alters the set of possible traces of the system models and the probabilities in such a way that no overall hazard probability can be computed.
Very interesting approaches are developed in the COMPASS [BCK+09a] project with its SLIM modeling language [BCK+09b] and in [GCW07]. Both approaches are similar to ours in analyzing whole system models. The difference is, that solely continuous time models are used. This is well suited to asynchronous interleaved systems but not to syn-

[4] Derived from the formal model of the railroad crossing.

[5] Consider for example that the faster the train, the more likely is the situation that the sensor is passed without sensor failure. A slower train speed translates to a higher probability of a $error\_passed$ failure mode, as the more time passes the more the probability shrinks that the failure does not occur.

chronous parallel systems [HKMKS00]. This also means that per-demand failure modes are not supported, as these are not directly expressible in the continuous time semantics.

5 Conclusion

The quantitative model-based safety analysis method presented in [GO10] was applied to a real world case study of a radio-based railroad crossing. The modeling included both per-time and per-demand failure modes and normally distributed deviations of a sensor. The limiting factor of the analysis method is the size of the state space (which is larger for probabilistic analyses than for qualitative analyses) and the running time of the model-checking. But as we have shown, a realistic case study is analyzable in that way and probabilistic model-checking is an active research area. Different abstractions to reduce the state space have been proposed, multi-terminal BDDs [KNP02] are already in use in PRISM, bisimulation- based abstractions are integrated in MRMC [KKZJ07] and the necessary numerical analysis can be moved to massive parallel GPU computation [BES09]. The analysis results are very promising, as all the stochastic dependencies are automatically reflected in the analysis, resulting from the analysis using probabilistic model-checking techniques. The different types of failure modes can be integrated into the model and although the system is modeled in a discrete time context, accurate continuous time approximation is possible. Future work will include developing a modeling framework for the combined usage of qualitative safety analysis, to compute the critical combinations and quantitative safety analysis to compute the overall hazard probabilities.

References

[ABB+06] O. Akerlund, P. Bieber, E. Boede, M. Bozzano, M. Bretschneider, C. Castel, A. Cavallo, M. Cifaldi, J. Gauthier, A. Griffault, O. Lisagor, A. Luedtke, S. Metge, C. Papadopoulos, T. Peikenkamp, L. Sagaspe, C. Seguin, H. Trivedi, and L. Valacca. ISAAC, a framework for integrated safety analysis of functional, geometrical and human aspects. In *Proceedings of ERTS 2006*, 2006.

[ADS+04] Parosh Aziz Abdulla, Johann Deneux, Gunnar Stalmarck, Herman Agren, and Ove Akerlund. Designing Safe, Reliable Systems using SCADE. In *Proceedings of ISOLA'04*. Springer-Verlag, 2004.

[BCK+09a] Marco Bozzano, Alessandro Cimatti, Joost-Pieter Katoen, Viet Yen Nguyen, Thomas Noll, and Marco Roveri. COMPASS project webpage. http://compass.informatik.rwth-aachen.de/, 2009.

[BCK+09b] Marco Bozzano, Alessandro Cimatti, Joost-Pieter Katoen, Viet Yen Nguyen, Thomas Noll, and Marco Roveri. Model-Based Codesign of Critical Embedded Systems. In *2nd International Workshop on Model Based Architecting and Construction of Embedded Systems*, pages 87–91. CEUR-WS.org, 2009.

[BES09] D. Bosnacki, S. Edelkamp, and D. Sulewski. Efficient Probabilistic Model Checking on General Purpose Graphics Processors. In C. Pasareanu, editor, *Proc. 16th International SPIN Workshop*, volume 5578 of *LNCS*, pages 32–49. Springer, 2009.

[BPRW08] Eckard Böde, Thomas Peikenkamp, Jan Rakow, and Samuel Wischmeyer. Model Based Importance Analysis for Minimal Cut Sets. Reports of SFB/TR 14 AVACS 29, SFB/TR 14 AVACS, Apr 2008. ISSN: 1860-9821, http://www.avacs.org.

[BV03] M. Bozzano and Adolfo Villafiorita. Improving System Reliability via Model Checking: theFSAP/NuSMV-SA Safety Analysis Platform. In *Proceedings of SAFECOMP'03*, pages 49–62. Springer, 2003.

[GCW07] L. Grunske, R. Colvin, and K. Winter. Probabilistic Model-Checking Support for FMEA. In *Proc. 4th International Conference on Quantitative Evaluation of Systems (QEST'07)*, 2007.

[GO10] Matthias Güdemann and Frank Ortmeier. Probabilistic Model-based Safety Analysis. In *Proceedings of the 8^{th} Workshop on Quantitative Aspects of Programming Languages (QAPL10)*. EPTCS, 2010.

[HKMKS00] Holger Hermanns, Joost-Pieter Katoen, Joachim Meyer-Kayser, and Markus Siegle. A Markov Chain Model Checker. In *TACAS '00: Proceedings of the 6th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pages 347–362, London, UK, 2000. Springer-Verlag.

[ISO09] ISO/WD 26262: Road Vehicles-Functional Safety, 2009.

[KKZJ07] Joost-Pieter Katoen, Tim Kemna, Ivan S. Zapreev, and David N. Jansen. Bisimulation Minimisation Mostly Speeds Up Probabilistic Model Checking. In *TACAS*, pages 87–101, 2007.

[KNP02] M. Z. Kwiatkowska, G. Norman, and D. Parker. Probabilistic Symbolic Model Checking with PRISM: A Hybrid Approach. In *TACAS '02: Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 52–66, London, UK, 2002. Springer-Verlag.

[KT02] J. Klose and A. Thums. The STATEMATE Reference Model of the Reference Case Study 'Verkehrsleittechnik'. Technical Report 2002-01, Universität Augsburg, 2002.

[Lad08] Peter B. Ladkin. An Overview of IEC 61508 on EEPE Functional Safety, 2008.

[OGR07] Frank Ortmeier, Matthias Güdemann, and Wolfgang Reif. Formal Failure Models. In *First IFAC Workshop on Dependable Control of Discrete Systems (DCDS 07)*. Elsevier, 2007.

[OR04] F. Ortmeier and W. Reif. Failure-Sensitive Specification: A Formal Method for Finding Failure Modes. Technical Report 3, Institut für Informatik, Universität Augsburg, 2004.

[ORS05] F. Ortmeier, W. Reif, and G. Schellhorn. Formal safety analysis of a radio-based railroad crossing using Deductive Cause-Consequence Analysis (DCCA). In *Proceedings of 5th European Dependable Computing Conference EDCC*, volume 3463 of *LNCS*. Springer, 2005.

[ORS06] F. Ortmeier, W. Reif, and G. Schellhorn. Deductive Cause-Consequence Analysis (DCCA). In *Proceedings of IFAC World Congress*. Elsevier, 2006.

[RTC92] RTCA. DO-178B: Software Considerations in Airborne Systems and Equipment Certification, December, 1st 1992.

[VDF+02] Dr. W. Vesley, Dr. Joanne Dugan, J. Fragole, J. Minarik II, and J. Railsback. *Fault Tree Handbook with Aerospace Applications*. NASA Office of Safety and Mission Assurance, August 2002.