Smart Railroad Maintenance Engineering with Stochastic Model Checking

Dennis Guck[1], Joost-Pieter Katoen[1,2], Mariëlle Stoelinga[1], Ted Luiten[3], and Judi Romijn[4]

[1]University of Twente, the Netherlands
[2]RWTH Aachen University, Germany
[3]ProRail, the Netherlands
[4]Movares Nederland, the Netherlands

Abstract

RAMS (Reliability, Availability, Maintenance, Safety) requirements are utmost important for safety-critical systems like railroad infrastructure and signaling systems, and often imposed by law or other government regulations. Fault tree analysis (FTA, for short) is a widely applied industry standard for RAMS analysis [1, 2], and is often one of the techniques preferred by railways organizations [3]. FTA yields system availability and reliability, and can be used for critical path analysis. It can however not yet deal with a pressing aspect of railroad engineering: maintenance. While railroad infrastructure providers are focusing more and more on managing cost/performance ratios, RAMS can be considered as the performance specification, and maintenance the main cost driver. Methods facilitating the management of this ratio are still very uncommon.

This paper presents a powerful, flexible and transparent technique to incorporate a maintenance aspects in fault tree analysis, based on stochastic model checking. We are able to analyze and compare different maintenance strategies (such as age-based, clock-based and condition-dependent maintenance) and their impact on reliability and availability metrics. Thus, we facilitate the trade off between cost and RAMS performance.

To keep the underlying state space of our state space small, we deploy two aggressive state space reduction techniques, namely compositional aggregation, and smart semantics. We illustrate our approach on several existing, large fault tree models in a case study from Movares, a major RAMS consultancy firm in the Netherlands.

Keywords: Dynamic fault trees, maintenance, availability, reliability, cost, recovery.

1 Introduction

Fault trees (FTs) are a wide-spread, and often preferred [1, 2], technique for RAMS analysis. A fault tree specifies the failure behaviour of a system in terms of its components. The leaves of the tree represent component failures; the gates indicate how component failures propagate through the system, and lead to system level failures.

Fault trees currently only support elementary maintenance aspects, e.g., in the simple repair rates model, all faults of a component are repaired with the same speed. The railway application context, however, demands highly advanced maintenance models supporting condition-dependent strategies for preventive, corrective, clock- and age-based maintenance, inspection, monitoring, etc. In case of multiple failures, decision strategies need to specify which component(s) to repair first. In addition, maintenance costs are pivotal. FTA cannot yet cope with these advanced maintenance aspects.

In this paper, we equip (dynamic) FTs with maintenance aspects, yielding a powerful framework that enables one to compare the effect of different maintenance strategies on the availability, reliability, and recovery speed. This is achieved in a number of steps: First, we extend the expressive power of FTs with maintenance aspects such as maintenance actions like repair or replacement of components.

Second, we provide a formal semantics of this FT extension. Our semantics provides an unambiguous meaning to each FT, and acts as a blueprint for analysis and tool-support. A crucial aspect of our semantics is its compositionality: the semantics of a complex FT is given as the combination of the semantics of its elements (i.e. gates and leaves). The main advantage is that reduction techniques, simplifying the component FTs to accelerate their analysis, can be applied in a component-based manner [4]. This drastically improves the scalability of the approach as it significantly reduces peak-memory consumption. The underlying mathematical model that we use for the semantics is input/output-interactive Markov chains (I/O-IMCs) [5]. An important contribution of the current paper is that we further exploit the reduction techniques by introducing *smart semantics*. Smart semantics, which can be seen as a refinement of the interface-based minimization technique that was successfully deployed in [6], automatically tailors the semantics of a FT element (i.e. underlying of I/O-IMC) to the context where it is used. So, rather than using a generic I/O-IMC model to represent a FT element, we obtain a smaller one depending on the surrounding FT gates.

Finally, we report on advanced analysis techniques that allow for determining a number of interesting FT measures such as expected lifetime (what is the expected time until a certain scenario occurs?), long-run average behaviours (which fraction of time is the system in a given good state?), and timed reachability (what is the probability to reach a certain scenario within a given deadline?) [7]. The key to determining these measures is the exploitation of stochastic model checking [8, Chapter 10], an innovative technique heavily used by hardware manufacturers. The scientific core of our work is the extension, refinement, and tailoring of SMC techniques to the analysis of I/O-IMCs.

We illustrate our approach by means of a study of determining the availability of

railway safety mechanisms of a railroad trajectory at one of the major crossing-points in the Netherlands between north-south and east-west railway traffic. We present the results using a prototypical software tool-chain and compare the impact of two different maintenance strategies.

2 RAMS and FTA for railways: state-of-the-art in practice

Railroad systems are bound by high performance standards. In the Netherlands for instance, the government poses severe financial consequences for the companies involved in railroad transportation and maintenance when train service is delayed or disrupted too often or for too long, or when other performance objectives are violated. These performance objectives, imposed by Dutch government, are stated as quantitative requirements on the number of delayed or cancelled train services.

Likewise, safety of passengers, train/maintenance personnel, residents and passers-by is crucial. Quantitative requirements are imposed by Dutch government on all safety-critical systems and subsystems, such as automatic train protection. Here, the engineering of systems includes a demonstration that the quantitative safety requirements are met. Without a convincing quantitative proof, new systems are not allowed to be implemented and used in the Dutch railroads.

In order to assess the performance of a railway system or subsystem, Fault Tree Analysis (FTA) is the method preferred by the Dutch Railinfrastructure manager ProRail [3]. FTA is a topdown, deductive approach to determine the causes for undesired events (failures) and calculate the expected performance (frequency of failures, total down time, etc.) [9]. In our experience with the application FTA to railway engineering projects, there are some limitations to obtaining trustworthy and truly useful performance expectancy results.

In particular, there is a strong relationship between preventive maintenance and failure rates on the one hand, and between corrective maintenance and the repair rate on the other hand: The likelihood of a failure occurring is closely related to the amount and quality of preventive maintenance being performed. For failure data that is being monitored, this is usually unknown. Moreover, preventive maintenance practice may vary, e.g. leading to more failures when replacement intervals become longer. Likewise, the repair rate is closely related to corrective maintenance and depends on spare parts, reachability of the components, repair or replacement strategies, etc. It would be very useful to explore these relationships and optimize on the balance between costs and performance for different preventive and corrective maintenance strategies.

There is currently no way in FTA to model neither the relationship between maintenance activities and the failure and repair rates, nor the costs involved. Tools for Reliability Centered Maintenance (RCM) do facilitate this, but have the drawback that no conditional relationship between basic events is possible (only OR gates).

Finally, FTA can be used to determine the expected performance when the numbers

for the basic events are known. However, in order to optimize performance and/or costs, it is very useful to perform the calculation with unknown parameters in basic events. In case of a tolerable hazard rate analysis, the analysis is performed in the reverse direction: the failure rate of the top event (an undesired safety incident) is a given quantitative requirement and none of the failure rates are yet known. By dividing the budget of the top failure rate over the tree structure, failure rates for the basic events may be induced which are then used as quantitative requirements for the engineering process of the functionality involved.

This paper provides significant steps to overcome these drawbacks: by incorporating maintenance aspects in fault tree analysis, we can explore the relationships between maintenance costs and system performance. Moveover, stochastic model checking approach is able to perform parametric analysis. This is currently not implemented in our DFT analysis tools, but is an important topic for further research.

3 Fault Trees and Dynamic fault trees

A fault tree (FT) is a tree (or rather a directed acyclic graph, since subtrees can be shared) describing the failure propagation of a system in terms of the failure of its components. The leaves of a FT are labeled with *basic events* and the non-leaves with *gates*. The root is called *top-level event*.

Static fault trees allow AND, OR and VOTING gates. *Dynamic* fault trees (DFT) [1] extend static ones with a number of gates, to model common patterns in reliability engineering: functional dependencies can be specified via the FDEP gate; spare management via the SPARE gate; and sequencing via the PAND gate. Thus, static fault trees model and propagate the occurrence of faults, and dynamic fault trees take into account also the order in which failures occur. In this way, DFTs allow more convenient and compact modeling of failure behavior.

3.1 DFT elements

Example. The DFT depicted in Figure 3 represents the (simplified) failure behaviour of a railroad level crossing. The level crossing consists of three subsystems: the sensors, the barriers and the controller. As indicated by the top-level OR-gate (named Level crossing failure), the level crossing will fail if either one of these subsystems fails. There are in total four redundant sensors. The sensor system fails if at least two out of the four sensors will fail, as is modelled by the 2 out of 4 VOTING gate (named Sensors fail). Further there can be a detection problem due to a disconnection of the cables. In that case, all sensors will become unavailable. This is modelled by the FDEP gate (named No detection): the trigger (named Disconnection) will cause the failure of all its dependent events ($\text{Sensor}_1, \ldots, \text{Sensor}_4$). Finally, the barriers fail if either the main and spare motor fails, modelled by the SPARE gate (named Motors), or if the switch and then a motor fails, modelled by the PAND gate (named Switching

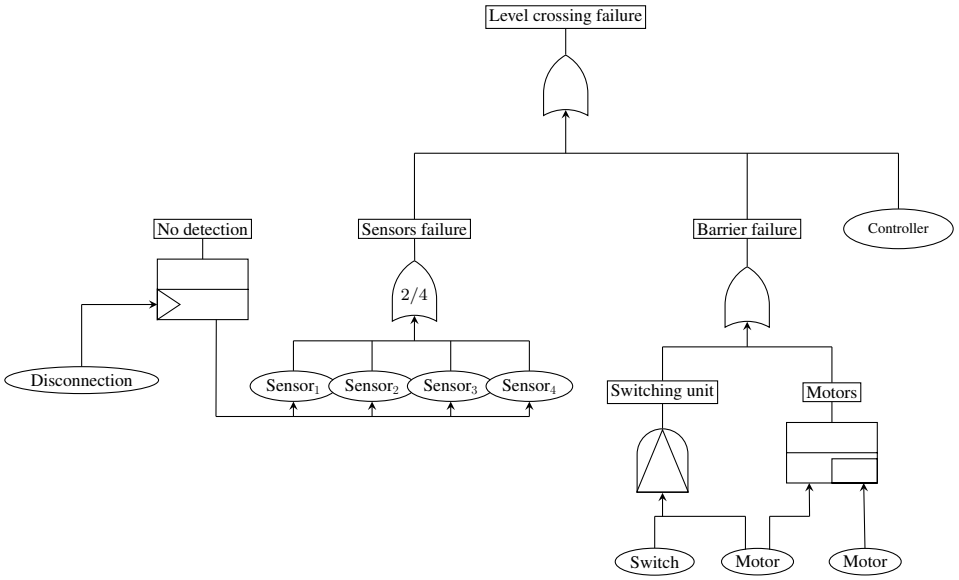
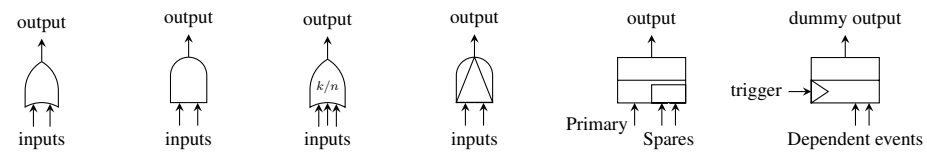


Figure 1: DFT for modeling a simple level crossing gate failure.

unit).

Basic events. A basic event (BE) represents the failure behaviour of a basic system component, and will be graphically displayed by a circle. A BE can be in three different modes: *dormant*, *active* and *failed*. The dormant mode indicates that a component is not in use, but acts as a stand-by or spare; the active mode indicates that the component is in use and is functional; the failed mode indicates that the component is out of order. In active mode, a component fails with a certain failure rate λ, which is the parameter of an exponential distribution. In other words, the mean failure time is given by $\frac{1}{\lambda}$. In dormant mode, the failure rate is decreased by a *dormancy factor* $\alpha \in [0, 1]$. In case $\alpha = 0$ the BE cannot fail (cold BE) and in case $\alpha = 1$ the failure rate is the same as in active mode (warm BE). Alternatively, the failure behaviour can be specified by a *phase-type* distribution. This is a probability distribution constructed out of several exponential distributions and is capable of supporting any actual distribution up to a given precision [10].



(a) OR (b) AND (c) VOTING (d) PAND (e) SPARE (f) FDEP

Figure 2: Dynamic fault tree gates.

Gates. A gate expresses how component failures induce a system failure. Gates consist of one or more inputs, and one output. Figure 2 depicts the DFT gates:

(a) The OR gate fails whenever at least one input fails.

(b) The AND gate fails whenever all of it inputs fail.

(c) The VOTING gate fails whenever at least k out of n inputs fail.

(d) The PAND (priority AND) gate fails whenever all of its inputs fail from left to right. It does not fail if some of its inputs fail in a different order.

(e) The SPARE gate consists of a *primary* input and one or more *spare* inputs. At system start, the primary is active, and the spares are in standby mode. When the primary input fails, one of the spare inputs is activated and replaces the primary. If no more spares are available, the SPARE gate fails. Note that a spare component can be shared among several spare gates.

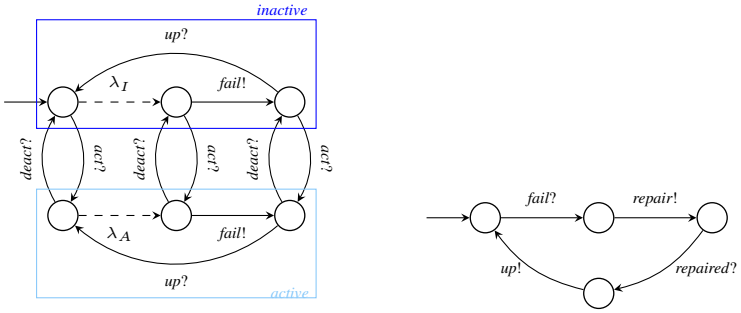
(f) The FDEP (functional dependency) gate consists of one *trigger* event and several *dependent events*. When the trigger event occurs, all dependent events fail. The FDEP has a "dummy" output, which is ignored in calculations.

4 Maintenance

Current FT methods [11, 12] — and their analysis techniques — are focused on the failure propagation of individual components until a top-level event is triggered. Although this aspect is of pivotal importance to RAMS analysis, the maintenance of system components and its influence on the system's failure behaviour is highly relevant. To that end, we introduce a DFT extension which captures maintenance activities such as component repairs. This is done in two steps: First we extend the behaviour of BEs, and then introduce a new module which handles the maintenance tasks. In the following we informally describe their behaviour.

4.1 Repairable basic events

In the standard DFT representation, once a BE fails, it is out of service and not considered any further. However, by repairing this BE and re-using it, the failure of the system could be temporarily prevented. Multiple repairs may further improve the system's availability. Such component repairs are not captured by the current DFT formalism. To include this functionality for BEs, we distinguish two modes: active and inactive. This distinction is important, since a failure of an activated BE can have a different impact on the system than a failure of an inactive component, e.g., they may incur different detection times until the failure of the BE is recognised. Further, after repairing the component it is important to know whether the BE was in active or in inactive mode to specify its role in the system. For example, consider a failure of a spare component. After its failure another spare component will be used if available.



(a) BE with inactive and active mode. (b) RM for a single BE.

Figure 3: Repair extension for maintainable components.

Thus, after the failed spare is repaired, it should change from active to inactive mode. On the contrary, if the spare was in inactive mode it should stay inactive.

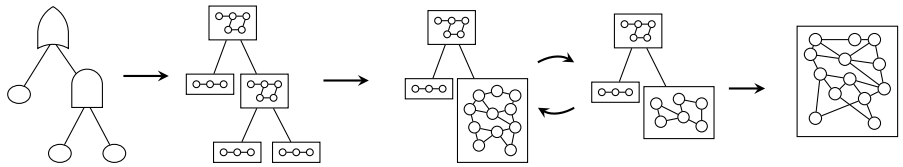
The incorporation of a repair functionality in BEs is done using a *repair module* (RM). The RM listens for the failure of the BE and on detecting a failure sends out a repair request. On receiving a message that a successful repair is carried out, the RM sends the BE an *up* signal indicating that the BE can be re-activated. This modularised architecture allows for a high flexibility in describing the components' maintenance procedure. Hence, depending on the defined procedure for a BE, we can plug in the corresponding RM. Further, in case the BE cannot be repaired we obtain the standard BE. A repairable BE and a simple RM are depicted in Figure 4.1 in terms of an I/O-IMC.

4.2 Repair unit

A so-called *repair unit* (RU) is exploited to determine the manner of repair as well as the repair's random duration. The activation of the repair process for a BE is triggered by the repair module RM of the BE. The RU then describes the actual maintenance procedure. After the RU has executed its process, it will send a *repaired* signal to the RM of the BE. To each BE a RU can be assigned. Hence, a RU will operate on one or more BEs. If more than one BE is assigned to a RU, the repair strategy — what is the order of repairing the failed components? — becomes an important factor. It is possible to directly model the RU with a certain strategy such as first come first serve. A more generic approach is to have non-deterministic choices, or assign priorities.

5 Semantics

In this section we describe our approach to model DFTs in a compositional and smart way. We first explain a more generic model, which can be used to include more expressiveness into the DFT and then describe optimizations regarding pattern matching in DFTs. For the formal encoding of the DFT semantics we use input-output interac-



(a) DFT (b) Transformation (c) Composition (d) Minimisation (e) IMC

Figure 4: Graphical overview of the compositional aggregation of DFT models.

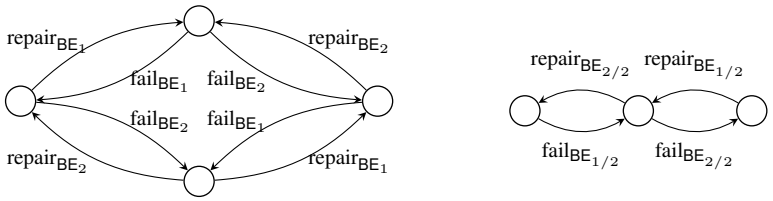
tive Markov chains (I/O-IMCs) which are an extension of interactive Markov Chains (IMCs) [5] by including input and output signals into the action set. An I/O-IMC consists out of a number of states which are connected via two types of transitions, *interactive* and *Markovian*. The interactive transitions are labelled with signals, e.g. "*fail !*" which is used for sending a failure signal to other components. The Markovian transitions are labelled with rates λ representing an exponential distributed delay with parameter λ.

5.1 Compositional aggregation

Compositional aggregation is an intelligent way of translating the DFT into an I/O-IMC while keeping the state space small. This method is exploited in the DFTCalc tool-chain [13], which is capable of computing several quantitative metrics on DFTs. Compositional aggregation, as depicted in Figure 4, starts with transforming each DFT element into an I/O-IMC. Then the obtained I/O-IMCs are iteratively composed in the following manner: Take two I/O-IMCs, compose them, hide all signals that are no longer needed for synchronisation, and then minimise the composition. This process continues until a single I/O-IMC remains. The order of the aggregation process heavily influences the maximal number of intermideate states, and is determined by a smart heuristic [14]. Remark that under different heuristics the resulting number of states at the end is the same. Compositional aggregation yields reductions upto several orders of magnitude [4] and in theory may yield exponential savings.

5.2 Smart semantics

In FT modelling a lot of redundancy is added to the FT to understand the systems behaviour. For example, instead of connecting all components leading to a system failure to one OR gate, they are grouped in several "OR-groups". This is important for the human perception of the FT, but for the analysis this behaviour can be simplified. Therefore, we introduce the notion of *smart semantics*. This is based on pattern matchings in the FT as well as finding equivalent behaviour of components. For example, if we analyse a FT with a group of independent OR gates, we can aggregate those gates into one OR gate for the quantitative analysis, respectively for AND gates. This holds also for independent FDEP gates, since those can be translated into OR gates.



(a) Naive way. (b) Smart semantics.

Figure 5: Comparison of 2 simplified BEs as IMCs.

Further, the aggregation of similar BEs will significantly reduce the state space. Considering a simplified BE it can be in two modes, active and failed. By considering now n BEs in parallel, we would obtain 2^n many configurations. If those BEs describe now the same failure and repair behaviour, we can reduce the possible configurations down to $n + 1$ by only considering how many BEs are already failed. Figure 5 depicts this for 2 BEs. Note that the treatment of gates that impose a certain order such as PAND and SPARE or and explicit counting as VOTING are more difficult to aggregate and are not considered at this point.

6 Analysis

Once the underlying I/O-IMC of a given extended DFT is obtained, the next step is to use efficient techniques for the quantitative evaluation.

6.1 Non-determinism yields bounds

As described above, the compositional semantics of DFTs yields an I/O-IMC. This model is a proper extension of continuous-time Markov chains (CTMCs), the model that is used for standard (mostly static) FT analysis. The major difference is the possible presence of non-determinism in IMCs. A CTMC is fully probabilistic, that is to say, being in a state the next state is determined by means of a random experiment. Technically this is due to the fact that CTMCs do only contain rate-labelled transitions, and no action-transitions. For (extended) DFTs, action transitions are natural representations of (repairs and) failure occurrences. As a result, non-determinism may occur, i.e., a state with multiple outgoing action-transitions. We stress that this phenomenon can also occur in standard DFTs, see [4], and thus as such is not a result of considering repairs..

The possible presence of non-determinism yields that we deal with stochastic *decision* processes. In fact, IMCs are very similar to continuous-time Markov decision processes (CTMDPs) [15]. In such models, probabilities of events are not uniquely determined, but instead subject to the resolution of the non-determinism. For instance, the probability to move to a certain state, e.g., the state in which the top-level element fails, depends on the non-deterministic repair strategy that is taken when more

than one component simultaneously has failed. The analysis of such models therefore yields upper and lower bounds. For any possible resolution of the non-determinism, the actual probability falls within these bounds. These bounds coincide if there is no non-determinism (as in the case study later on).

6.2 Quantitative measures

The quantitative analysis of I/O-IMCs (and thus maintenance DFTs) concentrates on three measures:

1. *Expected time*: what is the expected duration until some goal state is reached?
2. *Long-run average*: what is the fraction of time the DFT stays in a certain state in the long run?
3. *Timed reachability*: what is the probability to reach a certain state within a given deadline?

As stated above, all three analyses yield bounds, e.g., the maximal and minimal expected time. Example expected time objectives are: what is the expected duration between two successive failures of a switch?, or what is the expected time until a first failure of a BE? Long-run average measures are the pendant to stationary distributions of Markov chains. They are not of much interest for standard DFTs, as DFTs are acyclic, and thus will end up in a certain state, typically the failure of the top-level event, in the long run with probability one. For our extended DFTs though, components can be repaired, and long-run averages become highly relevant. Relevant properties are e.g., what is the fraction of time that several switches along a rail track have simultaneously failed? In our case study, we will cover several of such long-run average measures. Finally, timed reachability properties are of use for determining not just the expected time until a certain scenario occurs, but to check whether the probability that a certain event, e.g., the disruption of a railway crossing, happens within the next two days is at most 10^{-6}. Timed reachability probabilities are the natural counterpart to transient probabilities in Markov chains. The computation of long-run averages and expected time objectives relies on linear programming (plus some graph analysis) and can be done in polynomial time in the size of the IMC. Timed reachability is much more involved — like transient analysis of Markov chains has a higher time complexity than determining stationary distributions — and relies on discretisation. For details on the algorithms we refer to [7].

7 Case study

To demonstrate our approach, we analyse a set of fault tree models constructed by Movares to carry out RAMS analysis of a major railroad corridor. In this paper, we use anonymised data and altered fault tree models.

The main focus of the case study lies on the RAMS analysis on the railway safety systems of a railroad trajectory at one of the major crossing-points in the Netherlands between north-south and east-west railway traffic. The goal of the analysis is to verify that the rail trajectory fulfils the railway system specifications. Here, the focus lies on the availability of the systems on the rail trajectory. The failure occurrences are defined by three failure categories:

Category I. Severe disruption in both directions, such that no train can ride.

Category II. Severe disruption in one direction, such that no train can ride.

Category III. Minor disruption which leads to dis-punctuality.

Our experiments have been conducted on several DFTs, describing major functionalities of the safety system of the railroad trajectory. The analysis includes only rail-side safety equipment such as signalling, train detection, relay cabinets, high voltage cabinets and similar. Note that we use the following abbreviations in the fault tree descriptions:

- *ATB*: short for "automatische treinbenvloeding" which is a dutch automatic train protection system.
- *GRS*: which is a "General Railway Section Company" track circuit system.
- *PSSSL*: short for "prikspanningspoorstroomloop" which is a peak voltage track circuit system.

The analysis focuses on the availability of the safety system under two different maintenance strategies using our smart semantics approach. The results are listed in Table 1.

7.1 Fault trees

The FTs used in this case study are all static, hence only containing OR, AND and VOTING gates. This is due to the limitations of the currently used techniques in FTA as described in Section 2. Note that our method is capable of handling the PAND, SPARE and FDEP gates. It is future work, to obtain more succinct FTs of railroad models using these dynamic gates.

7.1.1 Category III failure

Failure in the authorization. This failure can be induced by several operational failures on the rail trajectory. Those are divided in five different failure events:

- The first event is induced by a signalling failure on the rail trajectory. This can be caused by any signal light failing or by failure of the cables.
- The second event is an unjust occupation. This failure is induced by wrong occupation signals due to a failure of the train detection.

- The third event is an impossibility to operate a switch, where it is in correct position. This failure is induced by a failure of the train detection.
- The fourth event is an unjust closing of a level crossing. This failure is induced by a failure of detection, a relay cabinet, or of a high voltage cabinet.
- The fifth event is a faulty ATB code. This failure is induced by a failure of the ATB functionality of a railway section.

The whole FT consists out of 40 BEs connected over 6 OR gates.

7.1.2 Category II failure

Failure of a switch. The failure of a single switch on the rail trajectory will block one rail direction and thus induces a category II failure. The corresponding FT consists out of three different OR events:

- The first event is the unavailability of a currently operated switch. A failure of a switch is again divided in three occurrences. The first occurrence is a failure on the rail trajectory on a currently used switch. The second occurrence is a failure in one of the stations on a currently used switch. The third occurrence is the failure of the cables in the station connected to a currently operated switch.
- The second event is the unreliability of a switch. This is similar to the first event, with the difference that all currently not operated switches are considered.
- The third event describes a failure of a release approval for shunting. This happens if either one of the control knobs fails or one of the cables fails.

The whole FT consists out of 27 BEs connected over 9 OR gates.

7.1.3 Category I failure

Failure of several relay cabinets. This failure is induced by a disruption of several relays on an operated track. This can happen by an actual defect of the relay cabinets, a defect of the high voltage cabinets or a defect in the power cables. A failure on the operated track will occur if either two different failures occur, e.g. a cable fails and a relay cabinet fails, or if two of the same category are failing, e.g. two high voltage cabinets are down. The whole FT consists out of 30 BEs connected over 4 VOTING gates and 4 OR gates. There exists also another FT with 66 BEs. An abstract illustration of this FT is given in Figure 6.

Failure of several switches. The failure of several switches will induce a category I failure. The FT lifts the failures of the previous described FT to the context of switches. Hence the failures leading to a failure of several switches are given by: The GRS detection is failing, or the PSSSL detection is failing, or a cable fails. The whole FT consists out of 26 BEs connected over 4 VOTING gates and 4 OR gates.

Disruption of a level crossing. The failure of a level crossing will induce a category I failure since the whole rail trajectory must be closed in this situation. The

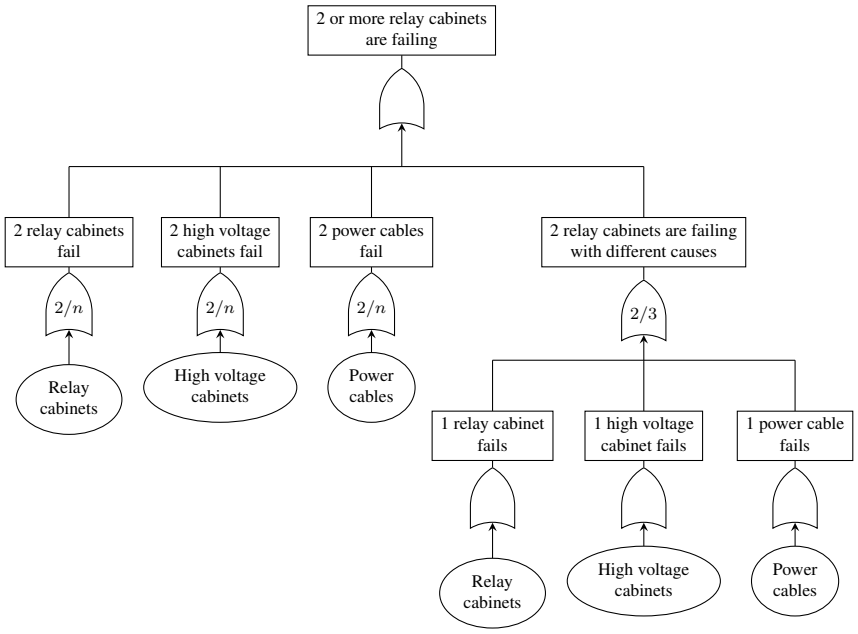


Figure 6: Illustration of one of the category I FTs.

level crossing will fail if the operating system fails as well as a wrongly activation of the crossing occurs. A wrongly activation can be induced by several detection or component failures. In total they are grouped in five events. The first event is a GRS detection failure. The second event is a PSSSL detection failure. The third event is an relay cabinet outage. The fourth event is a failure of one of the several cables. The fifth event is induced by a defect of the level crossing installation itself. All events can either occur in a station or on the midsection. The whole FT consists out of 58 BEs connected over 8 OR gates and one AND gate. There exists also another FT with 350 BEs.

7.2 Experimental results

We compute the models using two different repair strategies: in the first strategy, each component has a dedicated repair procedure, i.e., each component can be repaired at any time. This is the strategy Movares has considered for their analysis. A second strategy we consider is based on one repair per group of components. This is more realistic in practice: dedicated repairs are expensive, so in general repair units (personnel, spare parts, etc) are allocated to groups of similar components. Note that we consider an average repair time of 2 hours per component.

The results of our computations are listed in Table 1 and were conducted on a dual core Linux machine with 2GB of RAM. Note that strategy 1 refers to the model where all BEs have a dedicated repair and strategy 2 where a repair is assigned to a group of BEs. Further, the unavailability numbers are fractions, e.g. for the failure of a switch the system will be unavailable in 0.01926% of the time in the long run. By

System	# BEs	# States	Unavailability Strategy 1	Unavailability Strategy 2	Computation time
Failure in the authorization	40	12393	$2.8 \cdot 10^{-6}$	$2.8 \cdot 10^{-6}$	1 sec
Failure of a switch	27	90720	$1.926 \cdot 10^{-4}$	$1.926 \cdot 10^{-4}$	88 sec
Failure of several relay cabinets	30	46656	$8.71 \cdot 10^{-12}$	$1.2 \cdot 10^{-11}$	3 sec
Failure of several relay cabinets	66	2985983	$4.44 \cdot 10^{-11}$	$6.3 \cdot 10^{-11}$	6 min
Failure of several switches	26	14400	$4.44 \cdot 10^{-9}$	$7.15 \cdot 10^{-9}$	5 sec
Disruption of a level crossing	58	200447	$6.8 \cdot 10^{-8}$	$6.8 \cdot 10^{-8}$	18 sec
Disruption of a level crossing	350	11653631	$2 \cdot 10^{-7}$	$2 \cdot 10^{-7}$	30 min

Table 1: Results of the experiments.

observing the unavailability column of strategy 1 and 2, it stands out that the results only differ for 3 FTs. Further the variance is rather low. However, this shows that even with less possible repairs, the availability of the system is only affected marginally. In total, under both strategies, the system is available during 99.98% of the time in the long run. Thus, for the considered rail trajectory even a more relaxed maintenance strategy in terms of component repairs suffice to guarantee the same availability of the system. Note that the actual data of the FTs from Movares are confidential and cannot be displayed.

8 Conclusion

This paper presents an approach to integrate fault tree analysis and maintenance in the context of railroad systems. We presented a powerful and extensible technique based on SMC and their experimental validation.

Our technique paves the way for two important extensions, namely costs and parameters: Both are pivotal in facilitating the highly demanded cost/performance ratio for railroad management. Further the tailoring of SMC techniques and their integration in FTA is a natural next step.

Another important direction for further research is the automatic synthesis of optimal strategies. Our results show the applicability and usefulness of our general framework. However, the strategies used in this paper were still quite simple.

Acknowledgments

This research is financially supported by the STW-ProRail partnership program ExploRail under the project ArRangeer (12238), by the DFG/NWO bilateral project ROCKS (DN 63-257), and by the EU FP7 project SENSATION (318490).

References

[1] J. Bechta Dugan, S. J. Bavuso, and M. Boyd, "Dynamic fault-tree models for fault-tolerant computer systems," *Reliability, IEEE Transactions on*, vol. 41, no. 3, pp. 363–377, 1992.

[2] W. E. Veseley, F. F. Goldberg, N. H. Roberts, and D. F. Haasl, "Fault tree handbook, NUREG-0492," technical report, NASA, 1981.

[3] ProRail, "Leidraad voor RAMSHE - LMC-studie, HDL00032," tech. rep., 2010. (in Dutch).

[4] H. Boudali, P. Crouzen, and M. Stoelinga, "A rigorous, compositional, and extensible framework for dynamic fault tree analysis," *Dependable and Secure Computing, IEEE Transactions on*, vol. 7, pp. 128–143, 2010.

[5] H. Hermanns, *Interactive Markov Chains: And the Quest for Quantified Quality*. Berlin, Heidelberg: Springer-Verlag, 2002.

[6] S. Graf, B. Steffen, and G. Lüttgen, "Compositional minimisation of finite state systems using interface specifications," *Formal Asp. Comput.*, vol. 8, no. 5, pp. 607–616, 1996.

[7] D. Guck, T. Han, J.-P. Katoen, and M. Neuhausser, "Quantitative timed analysis of interactive Markov chains," in *4th NASA Formal Methods Symposium (NFM'12)*, vol. 7226 of *LNCS*, pp. 8–23, Springer, 2012.

[8] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[9] *Fault Tree Analysis*, vol. IEC 61025. International Electrotechnical Commission, 2.0 ed., 2006.

[10] R. Pulungan, *Reduction of Acyclic Phase-Type Representations*. PhD thesis, Universität des Saarlandes, Saarbruecken, Germany, 2009.

[11] H. Boudali, A. P. Nijmeijer, and M. I. A. Stoelinga, "DFTSim: A simulation tool for extended dynamic fault trees," in *Proc. of the 2009 Spring Simulation Multiconference (SpringSim '09)*, pp. 31:1–31:8, Society for Computer Simulation International, 2009.

[12] H. Boudali and J. B. Dugan, "A Bayesian network reliability modeling and analysis framework," *IEEE Transactions on Reliability*, vol. 55, pp. 86–97, 2005.

[13] F. Arnold, A. Belinfante, F. Van der Berg, D. Guck, and M. Stoelinga, "DFTCalc: a tool for efficient fault tree analysis," in *32nd Int. Conf. on Computer Safety, Reliability and Security (SAFECOMP'13)*, vol. 8153 of *LNCS*, pp. 293–301, Springer, 2013.

[14] P. Crouzen and F. Lang, "Smart reduction," in *Proceedings of the 14th Int. Conf. on Fundamental approaches to software engineering (FASE'11)*, vol. 6603 of *LNCS*, pp. 111–126, Springer, 2011.

[15] R. Knast, "Continuous-time probabilistic automata," *Information and Control*, vol. 15, pp. 335–352, 1969.