S3 Benchmark "A Fault Tree Based Model for ETCS Application Level 2"

AVACS S3*

[1] Albert-Ludwigs-Universität Freiburg, Fahnenbergplatz, 79085 Freiburg, Germany
[2] Carl von Ossietzky Universität Oldenburg, 26111 Oldenburg, Germany
[3] Universität des Saarlandes, 66041 Saarbrücken, Germany

Abstract. The *European Train Control System (ETCS)* strives for a safe and fast transnational railway service. ETCS application level 2 provides a continuous train speed supervision that protects against entering track sections already occupied by another train. Failures in the ETCS infrastructure as well as on the trains on-board unit may affect data integrity and thus yield safety critical situations.
In this case study, we exploit a fault tree description, given in the ETCS specification, to set up a STATEMATE model of ETCS level 2 in order to investigate novel techniques for analyzing *minimal cut sets* quantitatively.

1 General Description

ETCS and GSM-R (Global System for Mobile communications - Railway), an adaptation of GSM wireless protocol, are designed to replace the multitude of incompatible safety systems used by European Railways and enable safe fast transnational railway service. The role of ETCS to achieve safe railway service is twofold. First, information (e.g. speed or distance limits) is provided to the driver which has to be respected to maintain safety. Second, awareness to these informations is enforced. Unawareness may lead to an emergency brake. To meet the particularities of routes, different ETCS application levels were defined.

ETCS Level 2 In ETCS level 2 Movement Authorities (MAs) are granted by a track supervising radio block center (RBC) to a train. These authorize only one train at a time to enter a particular track section. The trains report their exact position and direction of travel at regular intervals to the RBC that hence can monitor train movements and grant (or deny) MAs accordingly. Together with an MA, the train receives also speed information and further route data.
Several failures in the overall ETCS system may finally bypass the supervising safety concept of ETCS. The driver, for example, will be allowed to drive to

* `http://www.avacs.org`

unsafe targets (e.g. to exceed the maximal permissible speed) without being protected by ETCS supervision, if the ETCS on-board unit received wrong targets itself. In case of a wrong (on-board) position determination, the position sent to the RBC will be wrong, too. Thus, the RBC could deliver wrong targets w.r.t. the actual train position. Hence, a particular instance for wrong position determination is a failure in the Balise Tracking Module (BTM), an ETCS on-board subcomponent: The train's ETCS on-board unit determines its position by sensors (axle transducers, accelerometer and radar) relative to track-side mounted Balises that effectively serve as reference points ("electronic milestones"). The Balises are also used to correct errors (i.e. inaccuracies) in the sensor based distance measurement. Thus, a failure in the BTM that is responsible for this correction, could yield a wrong position assumption.

The Context - Minimal Cut Sets The STATEMATE design was developed on the basis of a safety analysis described in terms of a fault tree given in the ETCS specification documents [4, 1–3]. This fault tree posed an ideal basis for the developed model, since in our ongoing work we are interested in analyzing the fault tree related construct of *minimal cut sets*. These may be considered as sets of failure scenarios sufficient and necessary to yield a given safety critical situation. Being able to extract such minimal cut sets, we exploit the *S3 tool chain* [6] to determine the minimal cut sets contribution to the probability of reaching a safety critical system state within a given time bound. This is achieved by (i) computing a minimal cut set specific system model and (ii) its subsequent analysis using the *S3 tool chain*.
At the current stage, the purpose of the described STATEMATE model is to demonstrate the reconstruction of the underlying fault tree and thus the extraction of the minimal cut sets. In the following, we (i) briefly sketch the S3 tool chain, that is, the verification environment, (ii) describe the STATEMATE model in terms of model components. Furthermore, we (iii) provide insight into the implemented failure scenarios and show how a failure may be captured in a Statechart.

1.1 S3 Tool Chain

For a detailed description of the S3 tool chain and the modeling formalisms being used, we have to refer the reader to [6]. An informal description of the tool chain is available in [7].

Timed Reachability Analyses The S3 tool chain enables to verify *timed reachability properties* of STATEMATE design based uniform continuous-time Markov decision processes (uCTMDPs). That is, we are able to ensure properties like:

"The probability to enter a safety critical system state within a mission time of 3 hours is at most 10^{-6}*."*

From the safety engineering point of view, the tool chain allows (i) to specify a complex system model in the high level language of Statecharts and (ii) to *enrich* this model by stochastic time constraints (i.e. delays). Technically certain Statechart transitions may be tagged in order to introduce stochastic delays that necessarily have to elapse between two transitions.
In terms of modeling formalisms this is achieved by the class of interactive Markov chains (IMCs) that are an orthogonal extension of labelled transition systems (LTS) and continuous-time Markov chains (CTMCs). While the *qualitative* STATEMATE design is transformed to an LTS, preserving the user specified Statechart transitions as labels, the stochastic delays are specified as Phase Type CTMCs that describe the time to absorption (i.e. the probability $P(D \leq t)$ that the delay D has expired at time t the latest). Both, the LTS and the stochastic delay CTMCs are composed to a monolithic uniform IMC that finally is transformed into a uCTMDP.
The key enabler to analyze timed reachability properties is a novel analysis algorithm for uCTMDPs that in particular has to resolve the *nondeterminism* inherent in uCTMDPs (and typically in the STATEMATE designs, too).

Stochastic Cut Sets Our approach of interweaving stochastic delays into a qualitative (nonstochastic) STATEMATE design enables the safety engineer to determine the system designs safety *quantitatively*, namely to determine the over-all probability, that is, considering all delays, of entering a safety critical system state.
The central idea is to exploit the S3 tool chain [6] to determine *minimal cut set* specific probabilities for reaching a safety critical system state within a given time bound. We basically partition the over-all probability of reaching a safety critical state in terms of relating each minimal cut set to a probability it (mainly) causes. It becomes possible to identify those components, whose failing contributes "most" to reaching a safety critical state. These are the components that should be replaced or improved first if the safety requirements can not be met. First, we have to determine the *minimal cut sets*, that is, to analyze the model and extract a fault tree. Second, we compute *minimal cut set* specific variants of the system model that retain the contribution of the failures in a particular *cut set*, while abolishing (most of) other failures contributions to the over-all probability. These variants are then passed to the tool chain.

2 Derived Model

In a nutshell, the model captures the following, basic ETCS level 2 scenario: The train's *On-board-Unit* (i) observes the speed by an *Odometer* and (ii) keeps track of occurring *Balises* by the *BTM*. These recurrently provide position information for the train. The position is sent to a track-side Radio Block Center (RBC) that informs the train about the current permissible speed and whether a movement

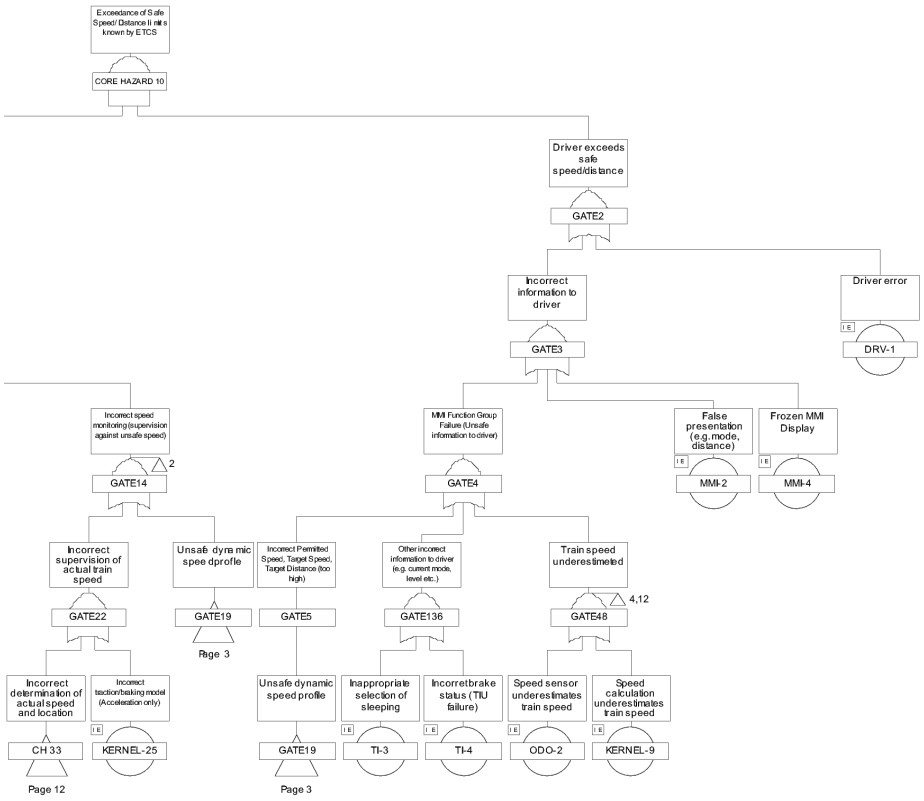

Fig. 1. Part of the Fault Tree given in [2]

authority (MA) for the next track segment is granted. Based on these informations, the trains *On-board-Unit* chooses a proper speed. The train's driver gets the same information and may set a proper speed, too. In our model, the minimum of these two speeds defines the train speed.
The over-all system reaches a safety critical state, if the train exceeds the current track segments permissible speed or enters a track segment without having received a valid MA. Such critical situations are caused by failures like an defective Odometer or an error in the trains *BTM*, as motivated above.
Having set the focus on the implementation of a STATEMATE model that comprises a set of failure scenarios to finally (re-)extract a fault tree, we followed the fault tree (cf. Fig.1) specification [2] and adjusted the model granularity accordingly. In particular, we had to balance the modeling effort and the models complexity on the one and the represented failure scenarios at the other hand. Hence, for example, we decided not to capture ETCS level- or mode changes in the model.

2.1 Statemate Model

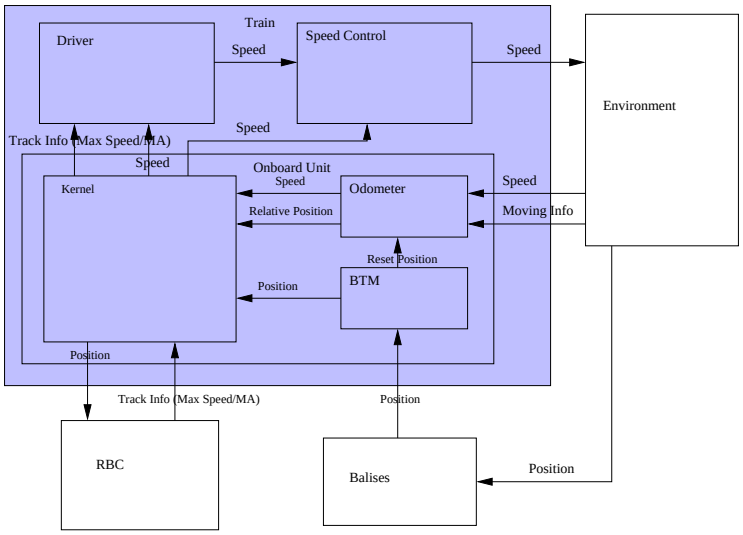

Fig. 2. Model Components.

Fig.2 shows the components captured in the STATEMATE design. It is partitioned in a *Train* (shaded blue) and three environmental components: The *RBC*, *Balises* and an *Environment*. The *Train* model contains an ETCS *On-board Unit*, a *Speed Control* and the *Driver* component, which abstracts the driver. These components implement the basic behavior specified for ETCS Level 2 in full supervision mode.
Note that we omitted ETCS on-board subsystems not being used in ETCS level 2, while others, listed in [4], are not modeled explicitly but integrated in other subsystems (e.g. EURORADIO with KERNEL, MMI with DRIVER, TUI with the representation of the *Physical Train*).
One noteworthy abstraction is that we did not model acceleration and braking behavior. The train drives forwards with speed `STAND`, `SLOW` or `FAST`. While an event "moving" with `FAST` speed already yields an increment of the trains position counter by one, two moves with speed `SLOW` do the same. The position is modeled as an integer $pos \in \{0, \ldots, 3\}$. This granularity is sufficient to integrate several failures of the model underlying fault tree.

Nominal Behavior The ETCS *On-board Unit* receives information about the current position and speed from its sensors, namely the *Odometer* and the *BTM*. The position is sent to the *RBC*. The *RBC* sends back *Track Info*, associated with the current track segment: In our model the maximum speed and a grant or denial of an MA. The information, the *Kernel* received from the *RBC*, is also

available for the *Driver*. With all this information, got from *RBC*, *BTM* and *Odometer*, the *Driver* and the *Kernel* decide what to do, that is, which speed to choose.
The minimum of the speeds determined by the *Driver* and the *Kernel*, is finally set as real train speed (i.e. target speed) by the *Speed Control* unit. That is, we did not implement an emergency break, initiated by ETCS here, but allow the ETCS (i.e. the *Kernel*) to overrule the *Driver* by setting a lower speed. The target speed is reported to the *Environment* that determines the new position (in dependence to the chosen speed) and sets speed and position for the *BTM* and the *Odometer*.
In ETCS level 2 the Balises are used as electric milestones. This is represented in the figure by the *Reset Position*, *Relative Position* and *Moving Info*. Each time the train moves (i.e. receives a *Moving Info* event) to the next position, the *Odometer* determines the position relative to the last occurrence of a Balise.

Time Consuming Nominal Behavior We use stochastic delays to model time consuming nominal behavior as follows.

Moving Every time a dedicated delay expires, an event is sent from the *Environment* and the train moves on depending on the current speed. If the current speed is `STAND`, the train does not move, if the speed is `SLOW` the train drives on a half position, and if the speed is `FAST` the train drives on a full position.

MA-Request, MA-Grant The grant of an MA to the train is delayed (w.r.t. to its request) by a dedicated stochastic delay. In particular, the train might have to wait for the grant due to a not yet cleared subsequent track segment.

These nominal behavior delays, associated to particular Statechart transitions, are not represented in the fault tree in [2], but they will be represented in our (re-)extracted fault tree (cf. section 3.1). Effectively, we take *all* user specified Statechart transitions into account for our (fault tree and stochastic cut set -) analysis that we use as synchronization points for the stochastic delays. Hence, we derive a fault tree that lists the specified transitions as building blocks.

Failure Behavior The fault tree in [2] informally describes different failure constellations that lead to the so called *'core hazard'*, safety critical states in terms of our timed reachability analysis. In the following, we describe how the failures affect the system's nominal behavior. We take the abbreviations used in the original fault tree ([2] pp. 11 to 38) to refer to the failure scenarios. An example for the concrete implementation of the described failures is given at the end of this section.
We implemented three failure scenarios that are crucial for a safe operation, that is, each of them can lead to the core hazard. A fourth failure (DRV-1), listed as safety related only in the ETCS specification, was implemented for experiments.

ODO-2 *"Speed sensor underestimates train speed"* If the actual speed of the train is `FAST`, the odometer falsifies the speed (nondeterministically) to `SLOW` or `STAND`. If the actual train speed is `SLOW`, the speed is set to `STAND`, respectively. Apart from exceeding the maximum speed, with this failure, it is also possible, that the train drives, although it has no moving authorization, because the driver and the kernel assume the train standing. In reality the driver should realize this, but such corrections are outside the scope of this model. It is not possible to repair this failure.

KERNEL-7 *"Incorrect last relevant balise group (LRBG)"* The *BTM* sends a wrong position to the *Kernel*, i.e. a random value. This yields directly a safety critical situation, as the position known by the *On-board Unit* is corrupted and therefore all information that will be received from the RBC will be invalid. Although the failure is modeled in the *BTM*, this is a *Kernel* failure, because the *Kernel* data is corrupted. In the model this data is placed in the *BTM*, to save state space. It is not possible to repair this failure.

TRANS-OB/RADIO-1 *"Incorrect Radio Message received by the on-board Kernel functions as consistent"* The message send next by the RBC is corrupted, but the kernel assumes it erroneously as consistent. The corruption can result either in an incorrect MA, or in an incorrect maximum speed for the current track segment. These corruption lead into a safety critical system state if the incorrect informations are less restrictive than the original information. In this case, the train drives too fast or without an moving authorization. When fired, this failure is active for the next RBC-message only. All following messages are correct, until the delay triggering this failure expires again.

DRV-1 *"Driver Error"* The driver starts to exceed the maximum speed. In this case, he sets the speed that is passed to the *Speed Control* to `FAST` if the maximum speed is `SLOW`, and to `FAST` or `SLOW` if no MA was granted for the current track. This single failure is not able to cause the core hazard (unless we twist the intended failures effect), because the speed is supervised by the *On-board Unit*. Only if both, the driver and the *On-board Unit* force a speeding, the train will exceed the maximum permissible speed, too. This failure behavior is implemented using two stochastic delays. One delay describes the time until the failure behavior begins, and the other the time until the behavior ends. Hence, the driver eventually realizes his error.

The effect of the considered failures is an inconsistency between the actual speed, actual position, actual maximal permissible speed or actual movement authorization and the corresponding assumed values that lead to an exceedance of the safe speed and distance limits.

The (ODO-2) failure for example is modeled as depicted in Figure 3. Basically, the nominal behavior of the *Odometer* component is modeled in the state `Correct_Speed_Estimation`. The transition `FM_ODO_2_BEGIN` is one of the synchronization (i.e. user specified) edges. That is, having computed the LTS for the STATEMATE design, we are able to delay the transition from the `Correct_Speed_Estimation` to the `Underestimate_Speed` state, that implements a mal-

functioning *Odometer* component as follows:
Intuitively the *Odometer* component is defective and underestimates the actual speed. Obviously this eventually yields an exceedance of the speed limit, as the *Kernel* and the *Driver* (indirectly) decide in dependence to the underestimated speed, how to adjust the train speed to follow the maximum speed. For example let the maximal permissible speed for a track segment be `FAST` and the maximal speed for the following track segment be `SLOW`. For the first segment the train speed is correctly set to `FAST` by the *Kernel* and the *Driver*. If the (ODO-2) failure occurs at the moment, when the train travels from the first track segment to the second, the actual speed is underestimated by the *Odometer*, and could therefore be assumed as `SLOW`. Both the *Kernel* and the *Driver* now assume, that the train is already driving with the correct speed (`SLOW`), and do not adjust the speed by breaking. This means the train drives on with `FAST` speed, and violates the maximal permissible speed.

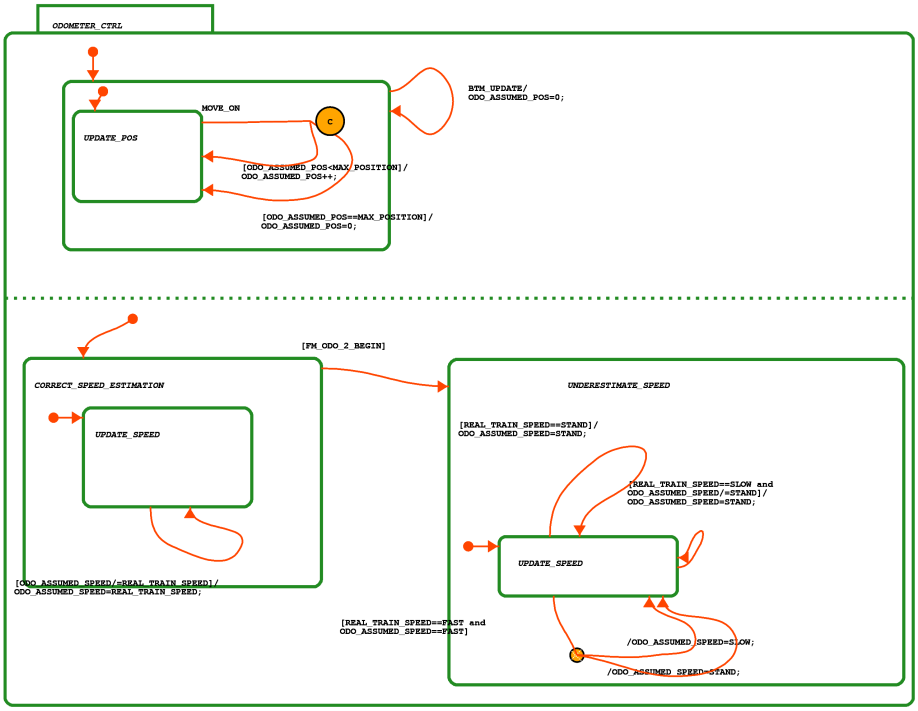

Fig. 3. Effect of the Odo-2 Failure

3 Interim Results

In the following we present numbers obtained from our tool chain. Furthermore, we present the re-extracted fault tree, which we are going to exploit in a second step of *minimal cut set* specific timed reachability analysis.

3.1 Timed Reachability Analysis Results

In this section, we give some statistics we obtained from experiments on the ETCS case study where we vary the failure scenarios as follows:

1. (ODO-2)
2. (KERNEL-7)
3. (DRV-1) and (ODO-2)
4. (DRV-1) and (ODO-2) and (KERNEL-7)
5. (DRV-1) and (ODO-2) and (KERNEL-7) and (TRANS-OB/RADIO1)

That is, we generate model variants (#1 - #5) that differ in the represented failure scenarios.
Being interested in stochastic fault tree analyses in the first place, we accepted inaccuracies in the concrete delay specifications. In particular, we estimated some of the stochastic delays on the basis of the numbers in [5] and synthesis thereof.
Table 1 gives an overview of the computation time and the model sizes for the symbolic part of our tool chain. We show the actual reachable states in the Stm2Lts computed LTS and the result of symbolic branching minimization, as generated by SigRef, as well as the overall computation time (in seconds).
In Table 2, we report results concerning the construction and minimization of the final IMC model. Experimental results are displayed for the compositional construction step, that is the time constraint weaving. The stochastic delays are composed to the system LTS one-by-one. Each construction step (i) generates an intermediate LTS and (ii) reduces its complexity by stochastic branching minimization. We report the size of the *largest intermediate state space* we needed to handle, the state spaces of the final results and the accumulated time (Generation and +Minimization) over all steps.
Statistical results for the transformation from IMC to CTMDP are displayed in Table 3. We give the number of states and transitions for the quotient IMC and the resulting CTMDP, together with the computation time required for this transformation. The column depicting the number of CTMDP transitions deserves a special comment. Since transitions in CTMDPs are triples (s, l, R) with a function R assigning rates to successor states, representing one transition may in the worst case already require space in the order of the number of states. Of course, this is not the case, the functions are very sparse. The numbers denoted in brackets are the average number of nonzero entries per transition.
The runtime of the extended MRMC model checker is shown in the last two columns of Table 3. The computation time needed to compute the worst case probability to reach the set of safety critical states has been computed for time bounds of 10 and 100 hours, respectively.

Table 1. Symbolic Steps: Statemate Safety Analysis and Minimization Statistics

Exp. #	LTS extraction			Branching Bisimulation		
	Reachable		Time	Min. Result		Time
	s	t	(sec.)	s	t	(sec.)
1	5876	7717	135.9	62	157	1.35
2	37380	55349	678.0	630	1701	21.58
3	22132	34069	224.0	206	682	5.43
4	149420	254413	1538.3	2272	7826	209.15
5	1062308	2045581	13686.6	13413	50205	3737.37

Table 2. Explicit Steps: Composition and Minimization Statistics

Exp. #	Phases	Compositional Construction			Final Quotient IMC	
		States	Transitions	G + M Time (sec.)	States	Transitions
1	1	123	383	4.73	15	35
	5	876	2658	4.73	403	1181
2	1	1259	4568	5.93	24	64
	5	4930	19612	6.66	2110	8118
3	1	536	2366	7.62	44	130
	5	16794	79392	23.25	10859	51330
4	1	4544	19702	10.18	141	415
	5	174425	961395	7425.31	146450	750791
5	1	17997	84290	16.18	442	1492
	5	1753166	11079825	238642.77	838888	5144242

Table 3. Explicit Steps: CTMDP Transformation and Analysis Statistics

Exp. #	Phases	Quotient IMC		Uniform CTMDP		Time	Time for Analysis of Formula (sec.)	
		States	Transitions	States	Transitions	(sec.)	$\sup_D \Pr_D(s, \overset{\leq 10}{\rightsquigarrow} B)$	$\sup_D \Pr_D(s, \overset{\leq 100}{\rightsquigarrow} B)$
1	1	15	34	25	43	0.36	0.01	0.12
	5	403	1180	793	1569	0.37	0.35	3.25
2	1	24	63	40	80	0.34	0.02	0.21
	5	2110	8117	4140	10143	0.47	2.01	19.05
3	1	44	129	65	141	0.36	0.03	0.32
	5	10859	51329	21279	61666	1.55	16.04	152.90
4	1	141	414	154	383	0.36	0.10	0.92
	5	146450	750790	247703	833119	18.83	301.09	2855.08
5	1	442	1491	559	1606	0.28	0.41	3.88
	5	838888	5144241	1570899	5856170	63.55	2563.62	24369.71

3.2 Extracted Fault Tree

Today, the model supports the failure scenarios listed in section 2. We are able to (re-)extract a fault tree as depicted in Fig. 5. Having discussed the failure effects and the nominal behavior delays above, in the following we will explain particular events in the fault tree. Note that we compute (and present) the fault tree in terms of LTS transition labels. Basically, the fault tree codes sets of paths to a safety critical state. The occurence of an event node in the fault tree means that the associated transition has to be observed at least once on a certain path.

Stable The model underlying fault tree in [2] describes different failure constellations that lead to the core hazard. Technically, in STM2LTS, we use a state predicate to define Statechart states representing a safety critical system situation. That is, entering of one of these states represents the core hazard's occurrence.

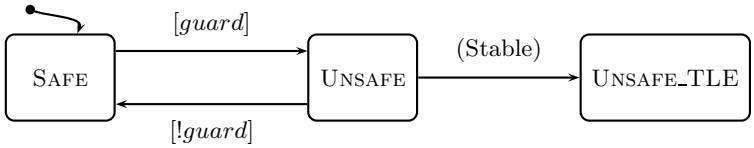

Fig. 4. The Observer.

We implemented an *Observer* (cf. Fig.4), that is, basically a watch dog Statechart that alternates between a `Safe` and an `Unsafe` state depending on the evaluation of a dedicated guard expression (and its negation respectively). The *Observer* enters state `UNSAFE`, only if the guard is satisfied, that is

1. no MA was granted and the train does not stand,
2. the actual train position is unequal to the position assumed by the on-board ETCS or
3. an exceedance of the maximal permissible speed was observed.

Unfortunately, this does not imply that the core hazard is reached, since there exist unstable configurations in the over-all Statechart executions that satisfy the guard too. (We borrow the notion of *stability* from the standard asynchronous STATEMATE semantics here.) For example: After the *Environment* has calculated a new position, and before this position is propagated through the `BTM` to the `Kernel` the actual position differs from the position assumed by the `Kernel` and therefore fulfills the guard, too.
To prevent these snapshots from being considered as safety critical, a third state `UNSAFE_TLE` in the *Observer* was introduced. It is only reachable via an additional transition (Stable). If the safety property, described above, becomes invalid before the system becomes stable, the state `SAFE` is re-entered (instead of entering the `UNSAFE_TLE` state). Hence, if and only if the Statechart state `UNSAFE_TLE` is

reached, the core hazard (in terms of [2]) is detected.
Using this modeling technique, we derive a neutral element in our fault tree analysis: The *(Stable)* transition is part of each *minimal cut set*.

MA-Request, MA-Grant, Moving Another noteworthy point is the existence of both failure and nominal behavior in the depicted fault tree. The *minimal cut set* specific variants, we are going to compute in our analyses will represent both classes, too. Hence the derived, minimal cut set specific probabilities, are expected to be more accurate in comparison to traditional techniques that allow to factor in failure rates only.

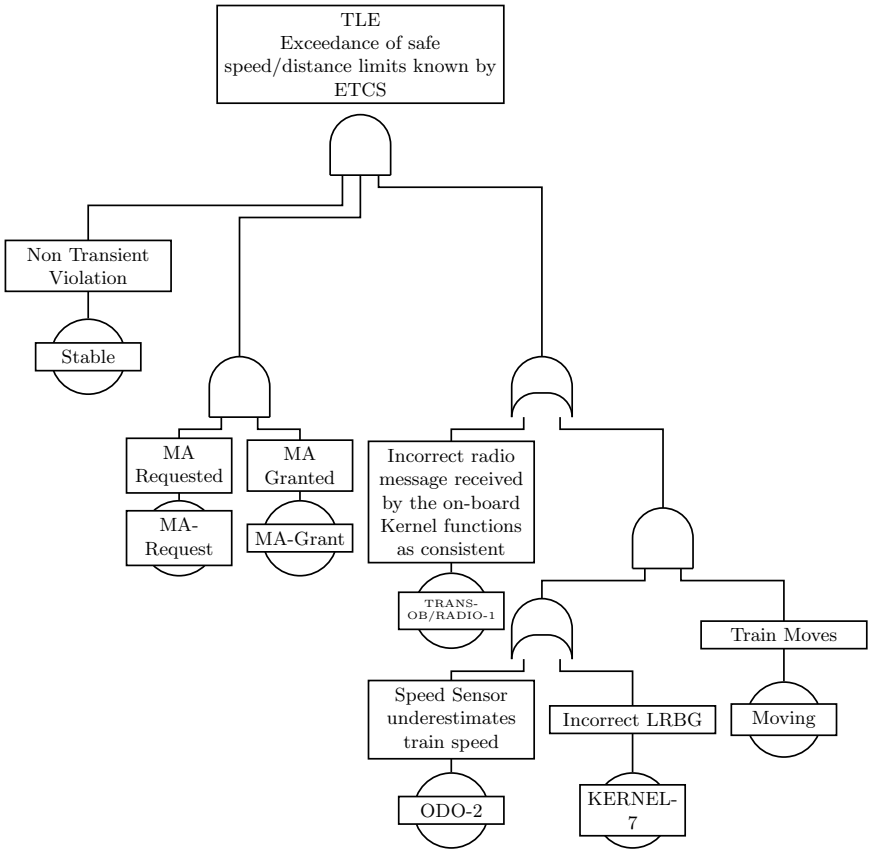

Fig. 5. Fault Tree computed for the STATEMATE model.

References

1. ETCS Application Level 1 & 2 - Safety Analysis - Part 0 - Document Overview. Technical report, ALCATEL, ALSTOM, ANSALDO SIGNAL, BOMBARDIER, INVENSYS RAIL, SIEMENS.
2. ETCS Application Level 2 - Safety Analysis - Part 1 - Functional Fault Tree. Technical report, ALCATEL, ALSTOM, ANSALDO SIGNAL, BOMBARDIER, INVENSYS RAIL, SIEMENS.
3. ETCS Application Level 2 - Safety Analysis - Part 2 - Functional Analyses. Technical report, ALCATEL, ALSTOM, ANSALDO SIGNAL, BOMBARDIER, INVENSYS RAIL, SIEMENS.
4. System Requirements Specification - Chapter 2 - Basic System Description. Technical report, ALCATEL, ALSTOM, ANSALDO SIGNAL, BOMBARDIER, INVENSYS RAIL, SIEMENS, 2002.
5. ETCS Application Level 1 & 2 - Safety Analysis - Part 3 - THR Apportionment. Technical report, ALCATEL, ALSTOM, ANSALDO SIGNAL, BOMBARDIER, INVENSYS RAIL, SIEMENS, 2005.
6. Eckard Böde, Marc Herbstritt, Holger Hermanns, Sven Johr, Thomas Peikenkamp, Reza Pulungan, Ralf Wimmer, and Bernd Becker. Compositional performability evaluation for statemate. In *3rd International Conference on the Quantitative Evaluation of Systems, QEST 2006, Riverside (USA)*, pages 167–178. IEEE Computer Society, 2006.
7. S3. S3 Benchmark 'Brake risk assessment for ETCS train platoons'. Technical report, AVACS.