# Reliability Analysis of Safety-Related Communication Architectures

Jan Peleska and Oliver Schulz

University of Bremen, 28359 Bremen, Germany,
{jp,oschulz}@informatik.uni-bremen.de,
WWW home page: http://www.informatik.uni-bremen.de/agbs

**Abstract.** In this paper we describe a novel concept for reliability analysis of communication architectures in safety-critical systems. This concept has been motivated by applications in the railway control systems domain, where transitions into stable safe state are usually considered as undesired events because they cause a severe deterioration of the service reliability expected by end users. We introduce a domain-specific language for modelling communication architectures, the protocols involved and the fault hypotheses about anticipated deviations of communication channels and possibly other components from expected behaviour. From such model, a generator creates mutant models associated with probability formulae expressing each mutant's probability of occurrence. Each mutant is analysed with respect to its unreliability, that is, whether it contains paths leading into stable safe state. Then the system reliability can be conservatively estimated by calculating an upper bound of the probability for the system to perform a transition into stable safe state within a given operational period. Our approach deliberately refrains from utilising probabilistic model checking, in order to avoid the state space explosions typically occurring when considering all possible erroneous behaviours within a single model. Instead, we analyse many different models, each only containing a restricted variant of deviations, which leads to faster evaluation times. In addition, several models can be evaluated in parallel in a distributed multi-core environment.

## 1 Introduction

### 1.1 Background: Safety Versus Reliability in Communicating Railway Control Systems

In safety related communication domains there are two important characteristics of communication architectures: Safety and reliability. In the railway domain the standard EN 50159-2 defines a basic design of communication architectures for safety related equipment. In general the standard splits the architecture into two parts: A safety layer, which must fulfil a specific safety integrity level (SIL) and a "grey channel" without any safety responsibility (see Fig. 1 and 2). Safety layers have to detect six different types of message errors to grant functional safety. The standard EN 50159-2 defines a defence matrix against these threads (Table 1,

[1, 2]). The safety reaction on such errors must be a safe state, which usually stops the communication service until the system is reinitialised or reset by an operator. Therefore a safe communication reduces the fault tolerance against arbitrary transmission errors and lowers the reliability of the communication architecture. To improve the fault tolerance against message errors it is necessary to use a reliable message transmission service (e.g. ARQ, Automatic Repeat Request) before the safety check is executed. A reliable transmission service can be included in the safety layer, in the upper protocol layer of the grey channel or in both layers (Fig. 2).

A "naive" combination of fault-tolerance mechanisms in the grey channel and safety layers will not necessarily increase the overall fault-tolerance: if, for example, lost messages in the grey channel lead to re-transmissions after timeouts, the message eventually passed to the receiving safety layer may be out-dated and therefore has to be discarded. As a consequence, it is necessary to perform analyses whether – given a trustworthy estimate for the occurrence of basic transmission faults as classified in Table 1 – the fault-tolerance mechanisms deployed in the grey channel will really increase the overall reliability of the distributed safety-critical control system.

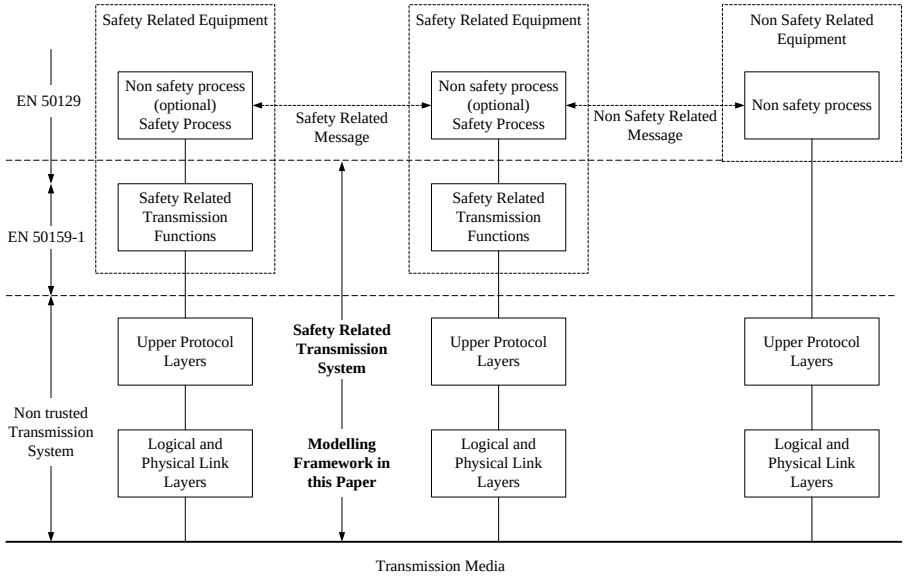


**Fig. 1.** Structure of safety-related communication architecture (from [2]). The term "Non Safety Process (optional)" in the *Safety Related Equipment* block indicates that also processes without safety-relevance can be deployed in the safety-critical equipment.

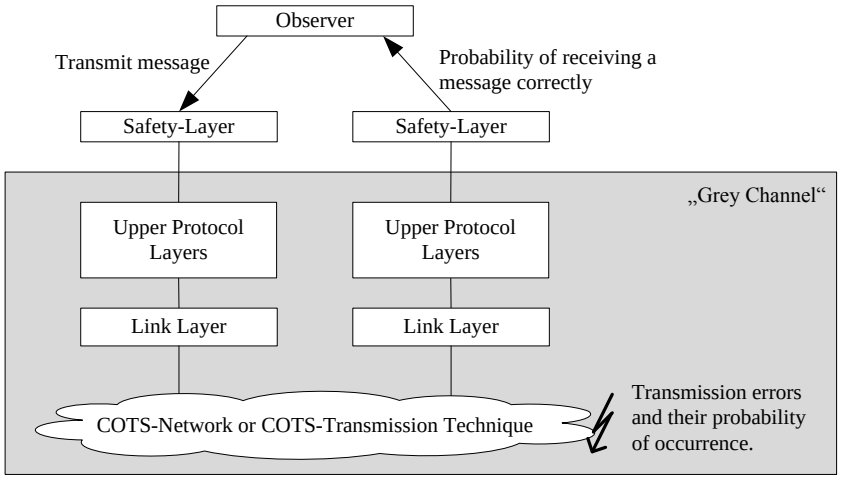


**Fig. 2.** General Modelling Architecture

**Table 1.** Threats

| Defences / Threat | Sequence number | Time stamp | Time out | Src. and dst. ID | Feed-back message | Identification procedure | Safety code |
|---|---|---|---|---|---|---|---|
| Repetition | x | x | | | | | |
| Deletion | x | | | | | | |
| Insertion | x | | | x | x | x | |
| Resequence | x | x | | | | | |
| Corruption | | | | | | | x |
| Delay | | x | x | | | | |

### 1.2 Objectives and Contributions

In this paper we present a novel method for reliability analysis of safety-related communication architectures structured into safety layers and grey channels as described in the previous section. In this context reliability is defined as the probability that the overall system will perform its (deterministic) safety-related services in a given operational time period $[t_1, t_2]$ without interruption and resulting transition into stable safe state, though transmission faults may occur in the grey channel with a given probability (see IEC 60050(191) [12] for the general definition).

Our analysis approach uses a domain-specific modelling language (DSL) developed by the authors. This DSL facilitates modelling communication architectures and protocols, together with the fault hypotheses concerning the probabilistic occurrence of the basic faults listed in Table 1. These communication models are used to create *mutants*, that is, derived models showing erroneous behaviour resulting from one or more basic faults occurring in compliance with

the fault hypotheses at various places in the communication architecture. For each mutant the probability of its occurrence can be calculated. Since the mutants themselves show deterministic (erroneous) behaviour, conventional non-probabilistic model checkers can be used to analyse whether the safety-related services will still operate properly in presence of the behaviour specified by the mutant. Time constraints play an important rôle in the behaviour of the system layers involved; therefore we have chosen Timed Automata [3] for modelling the mutant behaviour and use the UPPAAL tool [4] to perform the associated analyses. The verification goal $A\Box(\text{SAFE} \wedge \neg\phi)$ is to show that the safety layer will always satisfy its safety-specification SAFE and never transit into stable safe state $\phi$, despite of the faults occurring in the grey channel according to the mutant model under investigation. If a combination of faults on the grey channel leads to a violation of $A\Box\text{SAFE}$ the design has to be changed in any case, since a design-intrinsic safety violation that can be provoked by erroneous grey channel behaviour is not to be tolerated, regardless of the probability of its occurrence. If all mutants satisfy $A\Box\text{SAFE}$, they are classified by their occurrence probability, and according to their satisfaction or violation of $A\Box(\neg\phi)$. Then the resulting reliability of the overall system is calculated as the probability that only correct behaviour or mutants satisfying $A\Box(\neg\phi)$ occur during the given operational time period.

Our modelling approach requires *transaction-oriented* processing of safety-related communication functions: it is assumed that each activity consists of a bounded number of communication and processing steps, such that (1) the success or failure of the activity can be clearly determined after this sequence, and, (2) the success of the actual transaction is stochastically independent on the success of preceding actions. In the context of safety-related communication architectures this restriction is not a severe one: applications usually proceed according to different protocol phases like system setup, connection request, transmission of one application-specific datagram, and going through each of these phases corresponds to processing transactions of a specific type $T_\ell, \ell = 1, \ldots, q$. A minor limitation is discussed in Section 5.

We have developed an integrated tool chain starting with the modelling phase supported by the MetaEdit+ meta case tool [5] which was also used to design the DSL. A model-to-text generator creates an internal representation of the DSL model. A mutation generator creates the mutants from this model and calculates their occurrence probability. Each mutant is expressed by an XML text representation conforming to the internal input formal for UPPAAL models.

Our main contributions consist in the design of the DSL, the automated generation of the mutants and the calculation of their occurrence probability. Furthermore, our approach avoids the occurrence of state space explosions arising when all possible faulty behaviours are simultaneously considered in one probabilistic model (see further comments in Section 1.3). Finally, the different mutants can be analyses independently; therefore our analysis tool distributes the UPPAAL model checking tasks over several computers and CPU cores, so that model checking of different mutants can be performed simultaneously.

### 1.3 Related Work

Model-checking has been widely used for the verification of communication protocols and also for checking safety-properties of systems, see [6–8] and the references given there for related work in the railway domain. Reliability aspects have mostly been approached by means of probabilistic model checking, see, for example, [9, 10].

Our solution differs from the latter in that we deliberately do not use probabilistic model checking for these reliability aspects: extensive experiments performed by our group with the PRISM tool [10] showed that (1) the lack of real-time modelling capabilities enforces abstractions which either oversimplify the real communication behaviour or lead to unnecessarily complex constructions involving clock tick counters or similar devices, and (2) the incorporation of *all* possible faulty behaviours in one model lead to unacceptable checking times and even state explosions for the more sophisticated models. Indeed, since the probability that all possible faults occur while processing one transaction is so low that it can be neglected anyway, such a model would contain many computations of no practical relevance. Finally, (3) tools like PRISM only handle numeric probability values, but do not allow to investigate symbolic ones. As a consequence, parameter-dependent analyses require to re-run the time-consuming model checks for every parameter value to be considered.

Our approach tackles the combinatorial problem by checking many models instead of a single one and profit from the smaller size of each model: the complexity of evaluating one (probabilistic) model incorporating all possible faults is considerably higher than checking many simpler models, in particular, if the simpler models can be checked in parallel. Additionally, we calculate algebraic representations of ocurrence probabilities. As a consequence, parameter-dependent analyses can be made by just inserting concrete probability values into the parameters of the formula.

### 1.4 Overview

In Section 2 we sketch the work flow supporting reliability analysis and the tool components involved. Section 3 introduces the DSL CAMoLa, our description formalism for communication architectures. In Section 4 the principles of mutation generation and the reliability calculation based on mutant evaluation are described. Section 5 contains a discussion of results and prospects for future work.

## 2 Workflow and Tool Chain

The reliability analysis workflow starts with modelling a communication architecture in the domain-specific *Communication Architecture Modelling Language (CAMoLa)*, using the informal communication architecture specification with

associated protocol descriptions as input (Fig. 3). Next, CAMoLa's model-to-text generator transforms the CAMoLa model into an UPPAAL model, enriched with syntactic markers for the so-called *behaviour switches* which are part of the CAMoLa formalism and used to model possible deviations from normal behaviour (see Section 3 below). Now the mutation generator tool inserts *behaviour-vectors* (Section 3) into the UPPAAL model to create mutations with different message transmission behaviour. Intuitively speaking, each vector specifies which deviations from normal behaviour are applied to message sequences passing at specific locations in the model, and each model location where faulty behaviour is anticipated is associated with such a vector. The mutation generator records the algebraic formula for each mutation's occurrence probability in a table. Each formula is an arithmetic expression over the occurrence probability parameters associated with each fault type (see Table 1) possibly occurring in some part of the model when processing a message. Then the UPPAAL tool is activated to verify the reliability property on the mutation; this process is parallelised over several CPU cores and computers to increase performance. For each mutant, it is recorded in the table whether it shows reliable behaviour or leads to a transition into stable safe state.

## 3 The Communication Architecture Modelling Language CAMoLa

CAMoLa was designed for modelling communication architectures and associated protocol behaviours. Each model consists of synchronized processes representing protocol components, transmission channels or additional components simulating environment behaviour or acting as observers in the verification process. CAMoLa and its model-to-text generator were designed with the tool Metaedit+ [11], which is a meta-modelling and modelling-workbench [5]. The DSL supports two hierarchical views on communication architectures: A view on all components with their interactions (Fig. 4) and a process view on each component behaviour in timed automata notation (Fig. 5).

CAMoLa extends the usual timed automata notation by the notion of *behaviour-switches bs*, representing controlled normal and exceptional behaviour transitions between locations (see Fig. 5). Each possible controlled transitions is identified by a marker from set $o_{bs} = \{0, \ldots, n, stop\}$. The transition connected to one distinguished switch position (position 1 in Fig. 5) is associated with normal behaviour at this model location, so the error-free timed automata model can be extracted from the CAMoLa model by deleting at every behaviour switch all outgoing transitions but the one associated with normal behaviour. Each other switch position gives rise to a type of mutated behaviour.

In order to reflect the possibility of different types of transient errors occurring at a specific model location, mutant models are not simply generated from the CAMoLa model by fixing switch positions, but by associating each behaviour switch with *behaviour-vectors* $v^k$: if $o_{bs} = \{0, \ldots, n, stop\}$, then $v^k \in \{0, \ldots, n\}^k$, and it specifies that the first $k$ messages $m_1, \ldots, m_k$ passing along the model lo-

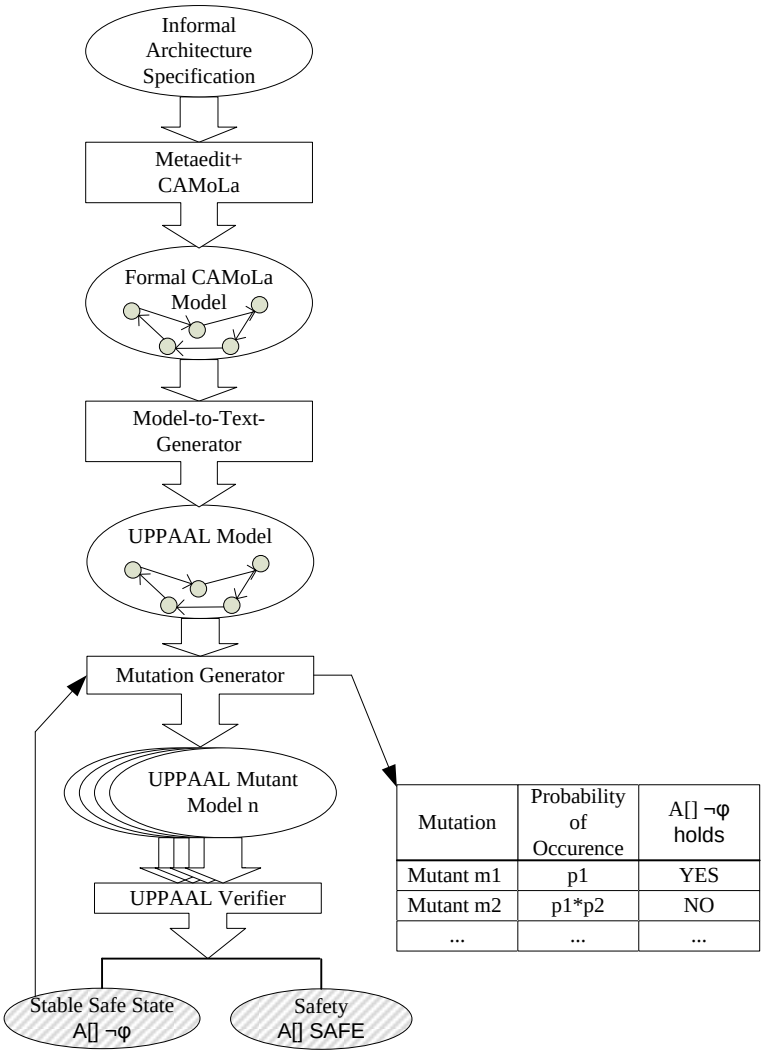


**Fig. 3.** Workflow of the presented Framework.

cation controlled by $bs$ trigger transitions $v^k(1), \ldots, v^k(k) \in \{0, \ldots, n\}$, respectively.

The semantics of this construction can be expressed by translating the CAMoLa process containing the pair $bs, v^k$ into an ordinary timed automaton utilising an additional auxiliary variable $i$ counting the number of messages passing along the behaviour switch, that is, the number of outgoing transitions of $bs$ which have been triggered so far, and an auxiliary location $l_{stop}$: suppose that $bs$ is located at source location $l$ and that the switch controls outgoing transitions with identifiers $0, \ldots, n$, leading to target locations $l_0, \ldots, l_n$. Then the associated timed

automaton has outgoing transitions

$$l \xrightarrow{i<k \wedge v^k(i)=0 \wedge i:=i+1;} l_0$$
$$l \xrightarrow{i<k \wedge v^k(i)=1 \wedge i:=i+1;} l_1$$
$$\vdots$$
$$l \xrightarrow{i<k \wedge v^k(i)=n \wedge i:=i+1;} l_n$$
$$l \xrightarrow{i \geq k} l_{stop}$$

at location $l$ ($i$ is initialised to 0 when the automaton is initialised).

While the designers specify the behaviour-switches and model the possible deviations from normal behaviour, behaviour-vectors are generated automatically by the mutation generator (Section 4). In order to control this generation process, each behaviour-switch position carries an upper bound indicating up to how many times the transition can be taken. The bound can be taken from the set $\mathbb{N}_0 \cup \{*\}$ (in the sample state machine of Fig. 5 only 0 and $*$ are used): symbol $*$ indicates that the mutant generator can select this transition an unbounded number of times when generating behaviour-vectors, a bound $m \in \mathbb{N}_0$ associated with transition $p$ constrains the behaviour-vector generation in such a way that $p$ occurs at most $m$ times in the vector.

Observe that all locations introduced on behalf of the behaviour-switch are urgent, since the switch is only a selector of normal or mutated behaviour, and does not consume processing time in the real world.

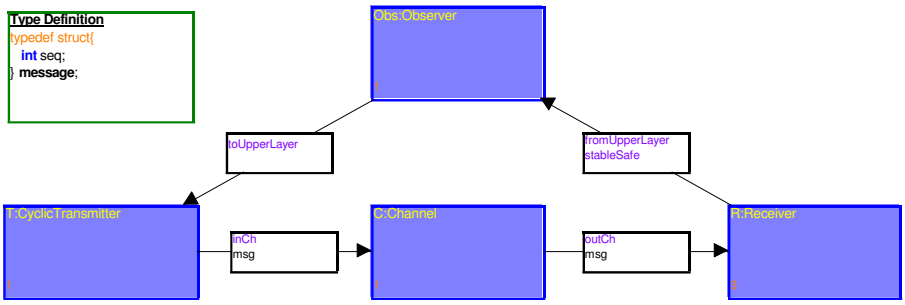


**Fig. 4.** Simple Architecture, System View

## 4 Mutation Generation and Reliability Calculation

**Generation Concept** Suppose we have created a CAMoLa model for each transaction type $T_\ell$ occurring in our communication architecture. The mutation

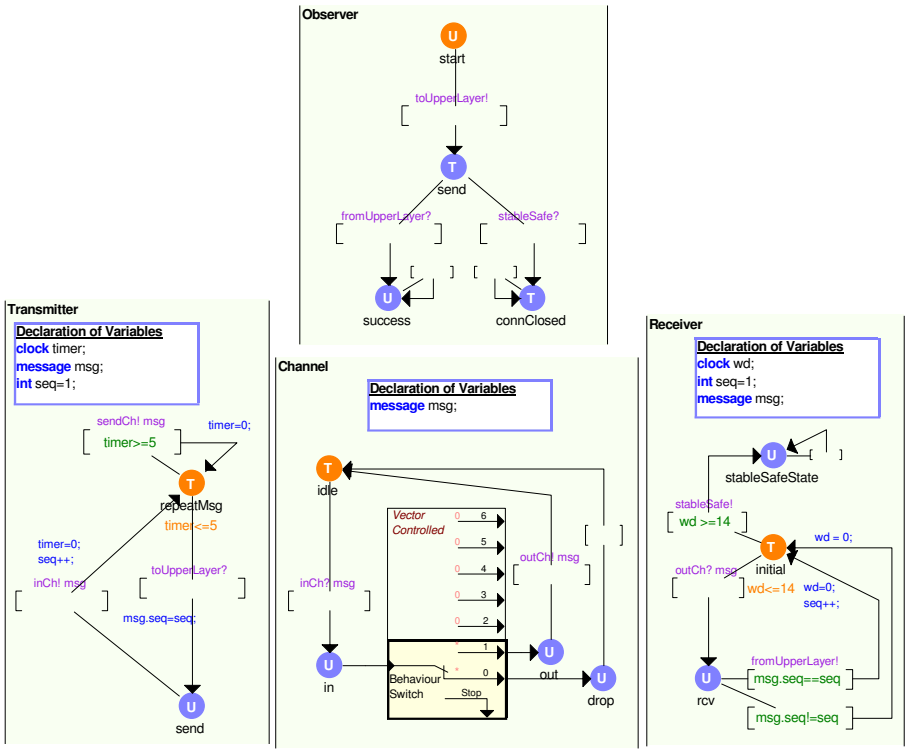


**Fig. 5.** Simple Architecture, Process View

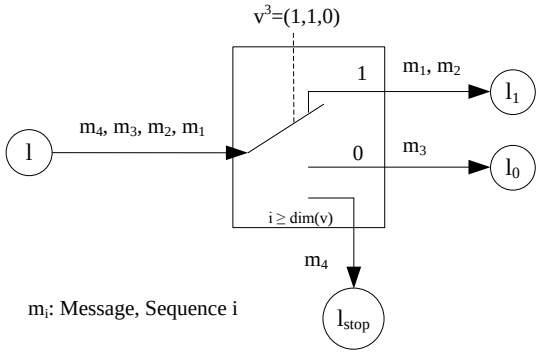


**Fig. 6.** Vector controlled Behaviour-Switch

generator creates concrete mutants as timed automata models, where all nondeterminism regarding fault occurrences has been eliminated. This is achieved by means of the behaviour vectors: let $\{bs_1, \ldots, bs_k\}$ the behaviour switches in the CAMoLa model associated with transaction type $T_\ell$. Given a bound $max \in \mathbb{N}$, the mutation generator creates tuples of behaviour vectors $V = (v_1^{d_1}, \ldots, v_k^{d_k})$, such that each behaviour switch $bs_i$ is associated with one behaviour vector $v_i^{d_i}$ of dimension $d_i$, and the following conditions are fulfilled: (1) $\forall i = 1, \ldots, k : d_i \leq max$, (2) each vector component $v_i^{d_i}(j), j = 1, \ldots, d_i$ is in range $\{0, \ldots, n_i\}$, such that an outgoing transition with identifier $v_i^{d_i}(j)$ exists at behaviour switch $bs_i$, (3) the mutants $\mathcal{M}(V)$ associated with $V$ satisfy $A\Box(\neg\phi)$ (we call them *reliable mutants*), and, (4) reducing the dimension of any $v_i^{d_i}$ by one will result in an unreliable mutant satisfying $E\Diamond\phi$. Conditions (3) and (4) are checked by means of the UPPAAL model checker.

**Calculation of Overall Reliability.** It is our objective to calculate an approximation of the communication architecture's expected reliability which is *conservative* in the sense that the real reliability is equal to or better than the calculated estimate. The calculations performed below are based on the assumptions that (1) no other faults occur in the communication system than the anticipated ones that have been represented in the CAMoLa model by means of behaviour switches, (2) all faults occur in a stochastically independent manner, and, (3) the safety-related services are performed in a transaction-oriented manner as explained in Section 1.2, so that the outcome of transactions is again stochastically independent.

If these hypotheses are satisfied it is possible to approximate the reliable system operation $R(t_0, t_1)$ over a time period $[t_0, t_1]$ by means of the reliability of single transactions: suppose that $R_{T_\ell}$ is the probability that execution of transactions of type $T_\ell$ will not transit into stable safe state, but perform the specified service, and that different transaction types $T_\ell, \ell = 1, \ldots, q$ have to be considered. For each transaction type $T_\ell$ let $c_\ell^{max} \in \mathbb{N}$ of the maximal number of $T_\ell$-transactions which are possible per time interval $[t_0, t_1]$, and $\delta_\ell > 0$ the minimal duration of such a transaction. Then the overall reliability $R(t_0, t_1)$ can be approximated conservatively by

$$R(t_0, t_1) \geq \min\{\prod_{\ell=1}^{q} (R_{T_\ell})^{c_\ell} \mid 0 \leq c_\ell \leq c_\ell^{max} \wedge t_1 - t_0 \leq \sum_{\ell=1}^{q} c_\ell \cdot \delta_\ell \leq t_1 - t_0 + \varepsilon\}$$

for $\varepsilon$ satisfying $0 < \varepsilon \leq \max\{\delta_\ell \mid 1 \leq \ell \leq q\}$. The right-hand side of the above formula represents the worst-case situation, where a maximal number of transactions is performed during time interval $[t_0, t_1]$, and the combination of transactions performed in this interval is technically still possible, but represents the least reliable combination which may occur.

It remains to determine the reliability of each transaction type $T_\ell$. To this end, we observe that the occurrence probability of a reliable mutant $\mathcal{M}(V)$ is

$$P_V = \prod_{i=1}^{k} \prod_{j=1}^{d_i} p^i_{v_i^{d_i}(j)}$$

where $p^i_m$ denotes the occurrence probability of the basic fault (or normal behaviour) associated with outgoing transition number $m$ at behaviour switch $bs_i$ (so $\sum_{j=1}^{n_i} p^i_j = 1$ for all $i = 1, \ldots, k$).

The probability that a transaction of type $T_\ell$ will terminate successfully without transition into stable safe state is

$$R_{T_\ell} = \sum_{\{V \mid \mathcal{M}(V) \models A\Box(\neg\phi)\}} P_V + \sum_{\{\pi, V \mid \mathcal{M}(V) \models E\Diamond\phi \wedge \pi \models \Box(\neg\phi)\}} P_\pi \cdot P_V$$

where $\pi$ denotes a computation of mutant $\mathcal{M}(V)$ and $P_\pi$ the probability of its occurrence: $R_{T_\ell}$ is the sum of all occurrence probabilities of reliable mutants plus the occurrence probabilities of paths in unreliable mutants leading to successful completion of the transaction. We neglect the occurrence possibility of the latter paths and only consider behaviour vectors of a maximal dimension $max \in \mathbb{N}$. This results in the conservative approximation

$$R_{T_\ell} \geq \sum_{\{V \mid \mathcal{M}(V) \models A\Box(\neg\phi) \wedge \forall i=1,\ldots,k: d_i \leq max\}} P_V$$

Since the right-hand side of this inequation can be computed by the mutant generator in combination with the model checker, this completes the conservative approximation for reliable system operation $R(t_0, t_1)$.

**Example.** As an example we demonstrate the calculation of the reliability of the example architecture in Fig. 4. This architecture consists of a transmitter, channel, receiver and observer, transmitter and receiver being allocated in the safety-layer.

The observer performs a safety-related transaction which completes successfully in terminal state success if a message sent on channel toUpperLayer is finally received on channel fromUpperLayer (see Fig. 5). The transmitter sends messages in fixed cycles of 5 time units. It repeats a message with the same sequence number until a next message has to be transmitted. The receiver removes duplicated messages indicated by equal sequence numbers. It also monitors the operability of the transmission channel: at least one message within 14 time units is expected (regardless of the sequence number). If no message is received within 14 time units, the receiver transits into stableSafeState, so we are interested in the probability that the complete system satisfies $A\Box\neg$Receiver.stableSafeState, or, equivalently, $RQ \equiv A\Diamond$Observer.success ("$RQ$" standing for *Reliability Query*).

The communication channel includes a behaviour-switch $bs_1$ with the set of outgoing transitions identified by $\{0, 1\}$. The outgoing transition number 0

models the message-loss-error and transition 1 transmits the message correctly. The *-character in the behaviour switch denotes that the transition can be take arbitrarily many times, so there are no restrictions regarding the creation of behaviour-vectors for $bs_1$.

The mutation generator generates the initial vectors $v^1_{1,1} = (1)$, $v^1_{1,2} = (0)$ and starts the model-checking processes to verify $RQ$ on each mutation. The model mutation induced by $v^1_{1,1} = (1)$ satisfies $RQ$, but the mutation induced by $v^1_{1,2} = (0)$ violates $RQ$, because the mutant derived from $v^1_{1,2}$ will drop the first message and block as soon as the second message arrives. In the next generation step, the mutation generator extends all vectors which are not satisfying $RQ$ by all possible outgoing transitions of the behaviour-switch – this results in $v^2_{1,1} = (0,1)$, $v^2_{1,2} = (0,0)$ – and starts again the verification process. The tool iterates until the dimension of the vectors have reached a predefined limit (in the example we set the limit to 4, because we know that $RQ$ can never be satisfied in presence of more than 3 message-drops). In Fig. 7 the whole set of generated behaviour-vectors is shown, each inducing one mutant model.

All behaviour-vectors whose mutants satisfy $RQ$ represent reliable computations of the communication architecture: each transmission where only fault-combinations still ensuring $A\Box\neg$Receiver.stableSafeState occur is still reliable. The probability of transmitting a sequence of messages specified in a behaviour-vector is calculated due to the known probability for an error-type to occur. In our example there is a probability to drop ($p_d$) or to transmit ($1-p_d$) a message. The probability that a sequence of controlled transitions occurs is the product of each transition probability in a behaviour-vector (e.g. $v^3 = (0,0,1)$, probability of occurrence: $p(v^3) = p_d \cdot p_d \cdot (1-p_d)$). We assume that all events are stochastically independent. Now the reliability of a communication model is the sum of all mutation-occurrence probabilities satisfying $RQ$. For the example system this results in the reliability formula $R_{Ex} = (1-p_d) + p_d \cdot (1-p_d) + p_d^2 \cdot (1-p_d)$ which can be reduced to $R_{Ex} = 1 - p_d^3$. □

## 5 Discussion and Future Work

The reliability analysis of communication architectures according to the concepts introduced in this article allows users to compare different architectural designs and fault-tolerance mechanisms of communication protocols in safety-related domains. Furthermore, the analysis results induce requirements on message error probabilities. These probabilities represent decision criteria whether specific transmission techniques like WLAN, IP-Networks or xDSL should be allowed or forbidden in safety-related communication architectures with high reliability requirements.

We have successfully analysed the reliability of the safety protocol SAHARA over UDP [13], a proprietary session-layer over the HDLC (High-Level Data Link Control) protocol, PROFIsafe over PROFINET and PROFINET DCP (Basic Discovery and Configuration Protocol). The results of these analyses imply maximal error probabilities and properties like maximal latencies of transmis-

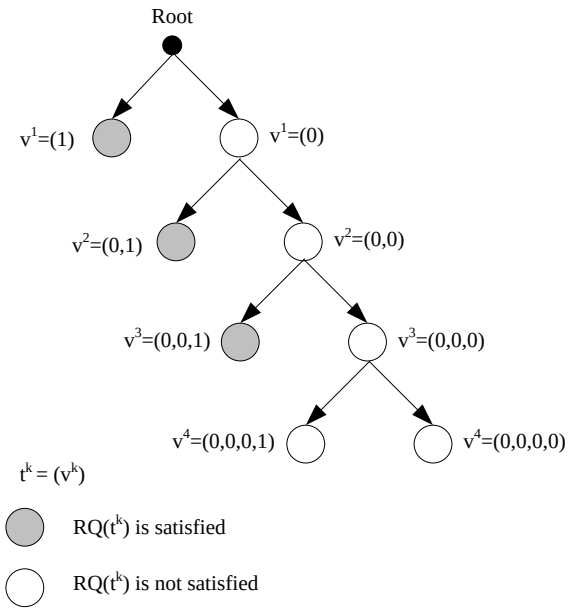


**Fig. 7.** Generated Vector Tree

sion techniques which are still acceptable in presence of the high levels of overall reliability required. Additionally the knowledge about the communication behaviour in presence of errors and error combinations has led to improvements of protocol specifications.

Due to the divide-and-conquer approach the availability of an array of computers and multiple CPU cores makes model checking feasible on a large amount of error combinations (i. e., mutants). We have successfully analysed an architecture with about 34 million error combinations which takes about 60 hours with an array of 25 computers (each 3 GHz).

In the future, we will analyse further architecture specifications, especially with reliable transport protocols like TCP and SCTP. Furthermore, we will improve the DSL CAMoLa for modelling communication architectures in a more generic way, such that pre-defined error behaviours applicable in specific communication domains can be re-used my means of building blocks from libraries. Additionally, it is planned to allow deviations from the transaction-oriented approach, in the sense that some system variables will be allowed to evolve across sequences of transactions. This will be helpful if, for example, fault counters are introduced in the system and incremented across transitions, so that shutdowns can be enforced if the fault rate is considered to be too high: in such a situation the success of a transaction also depends on the probability that the fault counter has reached its admissible limit before start of transaction.

**Acknowledgements.** The second author has been supported by Siemens AG in the context of the Graduate School on Embedded Systems GESy at the University of Bremen (`http://www.informatik.uni-bremen.de/gesy`).

**Originality.** *All necessary clearances for the publication of this paper have been obtained. If it is accepted, the authors will prepare the final manuscript in time for inclusion in the conference proceedings and will present the paper at the conference.*

## References

1. CENELEC: En 50159-1. railway applications -communication, signalling and processing systems part 1: Safety-related communication in closed transmission systems (2001)
2. CENELEC: En 50159-2. railway applications -communication, signalling and processing systems part 2: Safety related communication in open transmission systems (2001)
3. Alur, R., Dill, D.: A Theory of Timed Automata. Theoretical Computer Science (126) (1994) 183–235
4. Behrmann, G., David, A., Larsen, K.G.: A tutorial on UPPAAL. In Bernardo, M., Corradini, F., eds.: Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004. Number 3185 in LNCS, Springer–Verlag (September 2004) 200–236
5. Kelly, S., Lyytinen, K., Rossi, M.: Metaedit+ a fully configurable multi-user and multi-tool case and came environment. Advanced Information Systems Engineering **1080** (1996) 1–21
6. Esposito, R., Sanseviero, A., Lazzaro, A., Marmo, P.: Formal verification of ertms euroradio safety critical protocol. In: Proceedings of FORMS 2003, May 15-16, 2003, Budapest, Hungary. (2003)
7. Peleska, J., Große, D., Haxthausen, A.E., Drechsler, R.: Automated verification for train control systems. In Schnieder, E., Tarnai, G., eds.: Proceedings of the FORMS/FORMAT 2004 - Formal Methods for Automation and Safety in Railway and Automotive Systems, Technical University of Braunschweig (December 2004) 252–265 ISBN 3-9803363-8-7.
8. Schlingloff, F., Barthel: Verifikation und test des profisafe-sicherheitsprofils (2007)
9. Maxemchuk, N.F., Sabnani, K.K.: Probabilistic verification of communication protocols. In: PSTV. (1987) 307–320
10. Duflot, M., Fribourg, L., Hérault, T., Lassaigne, R., Magniette, F., Messika, S., Peyronnet, S., Picaronny, C.: Probabilistic model checking of the CSMA/CD protocol using PRISM and APMC. In: Proc. 4th Workshop on Automated Verification of Critical Systems (AVoCS'04). Volume 128(6) of Electronic Notes in Theoretical Computer Science., Elsevier Science (2004) 195–214
11. metacase.com: Metaedit+ workbench (2009)
12. IEC: Iec 60050-191-am1 ed1.0 amendment 1 - international electrotechnical vocabulary. chapter 191: Dependability and quality of service. (1999)
13. Kähloer, M.: The european train control system in thales signalling solutions. Mechanics Transport Communications **3** (2008) VIII–8–VIII–12
14. Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press (May 2008)