Dependability Checking with StoCharts: Is Train Radio Reliable Enough for Trains?*

David N. Jansen
Max-Planck-Institut für Informatik
Saarbrücken, Germany
dnjansen @ mpi-sb.mpg.de

Holger Hermanns
Universität des Saarlandes
Saarbrücken, Germany
hermanns @ cs.uni-sb.de

Abstract

Performance, dependability and quality of service (QoS) are prime aspects of the UML modelling domain. To capture these aspects effectively in the design phase, we have recently proposed StoCharts*, a conservative extension of UML statechart diagrams. In this paper, we apply the* StoChart *formalism to a safety critical design problem. We model a part of the European Train Control System specification, focusing on the risks of wireless communication failures in future high-speed cross-European trains. Stochastic model checking with the model checker* Prover *enables us to derive constraints under which the central quality requirements are satisfied by the* StoChart *model. The paper illustrates the flexibility and maturity of* StoCharts *to model real problems in safety critical system design.*

1. Introduction

The UML is pervading many challenging engineering areas including real-time and embedded system design. Embedded systems designers are usually facing various challenges if they strive for systems with *predictable quality of service* (QoS). Most QoS aspects of current embedded systems are time-related features and properties, and are of stochastic nature. A workable modelling and analysis approach to embedded system QoS is based on the observation that networks, interfaces, and even circuits on chips [10, 36] can be understood and modelled as discrete systems exhibiting some form of stochastic behaviour, such as error rates, response time distributions, or message queue lengths.

* Parts of this work are supported by the Special Research Activity SFB/TR 14 AVACS funded by the German Research Council DFG, by the Dutch Reserch Organisation NWO as part of the Vernieuwingsimpuls programme, and by the Dutch/German bilateral research initiative VOSS funded by NWO and DFG.

While in principle the UML provides the right ingredients to model discrete event dynamic systems, it lacks support for stochastic process modelling. This issue has been addressed in both the UML profile for schedulability, performance and time [38], and in the draft UML profile for modelling QoS and fault tolerance [37]. These profiles suggest annotational extensions of UML providing means to specify performance, dependability and QoS characteristics at various levels.

However, the vague semantics of the UML and of its annotational extensions drastically hamper QoS analysis. For this reason, quite some research has been devoted to formalise QoS analysis based on the UML. Several authors have suggested mappings of statechart fragments onto stochastic Petri nets variants [6, 23, 27, 28, 31]. Others have linked to process algebra [33, 9]. Together with Katoen [26] we have recently proposed a QoS-oriented extension of UML statechart diagrams, StoChart, which enhances the basic formalism with two distinguished features. One enhancement allows state transitions to select probabilistically out of different effects, much like the rolling of a die can have one out of six effects, determined probabilistically. The second extension is equally simple: The "after" operator of statechart diagrams is given a stochastic interpretation, allowing the use of arbitrary probability distributions for modelling, such as EXP[10 min] for a negative exponential distribution with a mean of 10 min, or UNIF[10 h,15 h] for a uniform distribution in the interval from 10 to 15 hours. The resulting statecharts dialect is called StoCharts, and contains basic UML statechart diagrams as a subset.

To make StoCharts a useful tool in QoS modelling and prediction requires more than a simple extension of UML with an intuitive interpretation. In order to support model-based QoS prediction, a formal mathematical interpretation is indispensable. Otherwise, model-based calculations are not trustworthy. Concretely, StoCharts are equipped with a rigid formal semantics. The semantics exploits lessons learned in a decade of research in formal

specification of stochastic processes, rooted in the seminal works on stochastic Petri nets [3, 2], stochastic process algebra [20, 22] and probabilistic automata [35]. In order to associate a stochastic interpretation to collaborative collections of statechart diagrams embedded in arbitrary environments, STOCHARTS are equipped with a compositional semantics, which uses concepts from Input/Output (I/O) automata [30]. The semantics associated with STOCHART is based on the requirements-level semantics of Eshuis and Wieringa [17], which adapts the Statemate semantics [21] and its formalisation by Damm et al. [11] to the UML. We are convinced that a different base semantics could have been chosen, as our extension is more or less orthogonal to the concepts of basic statechart diagrams.

This paper exemplifies the usefulness and maturity of STOCHARTS on a nontrivial case study. We apply the STOCHART formalism to model a safety critical fragment of the future European Train Control System (ETCS) standard. Communication among ETCS components (trains, trackside equipment etc.) will be based on mobile communication using GSM-R, an adaptation of the GSM protocol to railway applications.

The safe and efficient operation of ETCS is, of course, of prime importance. The specifications of GSM-R and of ETCS contain various QoS requirements such as *"a connection must be established within 5 seconds with 95 % probability"*. Due to the architecture of ETCS, on-board and trackside data processing as well as the radio communication link are crucial factors in ensuring the ETCS requirements. In order to study this issue, we follow an approach inspired by the *design-by-contract* paradigm [32, 7]. We treat the GSM-R specification as a black box in our model, only considering the QoS requirements specified for GSM-R, which turn into worst-case QoS *guarantees* for the service provided by GSM-R to the ETCS. These guarantees are part of the STOCHART model we develop. Our analysis, then, enables us to identify bounds on the distance between consecutive trains on a track, under which the crucial QoS requirements of ETCS are still satisfied. The computations are performed with the model-checker PROVER [39] which implements a variation of discrete event simulation to analyse the stochastic process underlying the STOCHART.

The STOCHART model of the ETCS communication fragment developed in this paper is not ultimately complex, and it can be extended in many interesting directions. We believe, however, that it shows how STOCHART can be used in a systems engineering context together with model checking-based dependability analysis. Our case study is inspired by a DSPN model proposed by Zimmermann and Hommel [40]. Our work differs from the one considered in [40] in a number of issues, besides the use of different modelling formalisms. In particular, we provide a more faithful modelling of the worst case GSM communication characteristics, and allow for concurrent (or simultaneous) faults of different types. On the analysis side, we use approximative stochastic model checking, as opposed to a numerical analysis of the stochastic process underlying the DSPN. As our (more faithful) model contains multiple concurrent non-exponential distributions, it is not obvious how it could be analysed numerically.

In a nutshell, the central contribution of this paper lies in the application of STOCHARTS to a realistic, yet abstract, modelling problem, where stochastic requirements and guarantees are assessed using stochastic verification technology.

Organisation of the paper. Section 2 introduces UML statechart diagrams and STOCHARTS, briefly touching upon semantic issues. Section 3 introduces the ETCS problem domain, and describes the particular train radio signalling case study. The STOCHARTS model and its analysis are presented in Sections 4 and 5. Section 6 discusses lessons learnt from this case study and concludes the paper.

2. STOCHARTS

This section gives an overview on UML statechart diagrams and STOCHART, and reviews the semantic model underlying STOCHART. We refer to [26] for a thorough discussion of the STOCHART formalism.

UML statechart diagrams. A basic UML statechart diagram consists of

- a set of *Nodes*[1] with a tree structure. Nodes are of type 'basic' (leaves), 'and' , or 'or'. The root node and children of *and*-nodes are of type 'or'. Each *or*-node has a distinguished, *initial* child node.
- a set of *Events* with the distinguished element $\perp$ denoting "no event required".
- a set of (typed) *Variables* or attributes together with an initial valuation $V_0 : \mathit{Vars} \to \mathcal{D}$, assigning initial values to the variables. Here $\mathcal{D}$ subsumes the domains of all variables.
- a set of *Edges*. Each edge connects a set of source nodes to a set of target nodes, and is labelled with an event, a guard (a Boolean expression), and a (possibly empty) set of actions (assignments to variables or sending messages to other objects).

For analysis purposes, we assume that all these sets are finite.

1 The UML specification for statechart diagrams actually speaks of *states,* but we prefer to call them otherwise because the system state may consist of more than one node at the same time.

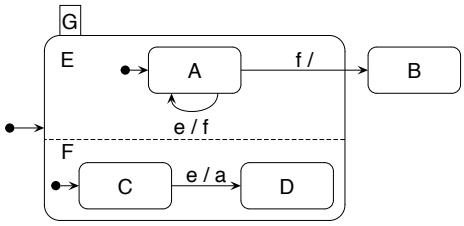

Figure 1. Example statechart diagram

Intuitive semantics of UML statechart diagrams. The statechart diagram is always in some state which consists of one or several nodes: if an *and*-node is part of the state, *all* of its children are in the state. If an *or*-node is part of the state, exactly *one* of its children is so. An edge may be taken (i. e., is enabled) if all its source nodes are part of the current state, its guard holds, and its event happens. The system selects as many enabled edges as possible for execution; a choice between conflicting edges is often resolved by the UML priority scheme. Once the selected edges are taken, their source nodes are left, their actions are executed, and their target nodes are entered. Note that multiple edges may be taken in a single step. To simplify the analysis, we assume that the step is instantaneous, so that no new events will reach the system before the step finishes.

Figure 1 contains a small example statechart diagram. The system shows an *and*-node, labelled G, with two parallel components, labelled E and F. While the system is in G, it is always in E *and* F. Each of these is an *or*-node. While the system is in F, it is always in either C *or* D. Arrows indicate edges.

For this example, the semantics is as follows. The initial state consists of the nodes A, C, E, F and G. Upon receiving event e, nodes A and C are left, actions a and f are generated, B is entered and A is re-entered. When both events e and f are received at once, there is a conflict: both edges leaving node A are enabled, but only one can be taken. (The UML semantics does not prescribe which one takes priority.)

Additional UML statechart diagram features. In addition to the above, the UML defines many further features of statechart diagrams. For example, there is an after(t) operator which can be used as a trigger for an edge; the operator indicates that the edge should be taken t time units after its source node is entered. Further, an *or*-node may contain a *history connector* meaning that the statechart should memorise which of its child nodes was active last and reenter that child node (instead of the initial one) in case the *or*-node is reentered. Finally, we will also use *entry* and *exit actions,* i. e. actions that are executed whenever some specific node is entered or exited. These actions are denoted with the labels on entry and on exit in the respective node. This and other features can be defined as syntactic sugar. We have not included them in the above definition for simplicity.

STOCHARTS. STOCHARTS extend UML statechart diagrams as follows: the "after" operator is extended to indicate random delays, and the notion of edges is refined into probabilistic edges (P-edges) to handle discrete probabilistic branching.

- Like an event, the after-operator acts as a trigger of an edge (or a P-edge). It has as parameter a cumulative distribution function F where $F(t)$ is the probability to wait at most t time units.
- A P-edge generalises an edge such that the set of target nodes and the actions can be chosen probabilistically. The trigger is the same as for an edge: it has a set of source nodes, an event or or an after operator, and a guard. Its reaction, however, is defined by a function assigning probabilities to pairs, consisting of a set of actions and a set of target nodes.

The UML notion of an edge is retained by a trivial P-edge with a single probabilistic outcome (having probability 1). A (nontrivial) P-edge is graphically depicted in two parts: an arrow labelled with an event and a guard directed to a P-pseudonode (denoted Ⓟ) from which several arrows to target nodes emanate, each labelled with a probability and an action set. For semantic reasons, we impose a slight restriction on P-pseudonodes: most arrows emanating a P-pseudonode that cross node borders are forbidden to simplify the probabilistic choice (see [25, 24] for an extensive discussion of this issue).

Intuitive semantics of STOCHARTS. Like a UML statechart diagram, a STOCHART is always in some state. A P-edge is enabled if all its source nodes are part of the current state, its guard holds, and either its event happens or the delay associated with its after operator expires. The system selects as many enabled P-edges as possible for execution and resolves the discrete probabilistic choices. Probabilistic choices of P-edges are assumed to be stochastically independent. Once the selected edges are taken, their source nodes are left, their actions are executed, and their target nodes are entered. On entering a node with an outgoing (P-) edge labelled with an after(F) operator, a sample is taken from distribution F and a timer is set accordingly. The corresponding outgoing edge becomes enabled once the timer expires.

Figure 2 shows a small example of a STOCHART. Node H is the initial node. In this node, the system waits a random time, with a mean of 10 seconds, exponentially distributed. After that, it proceeds to node K and its default child I. In this state, the system reacts to trigger a by moving to node J with probability 0.1.

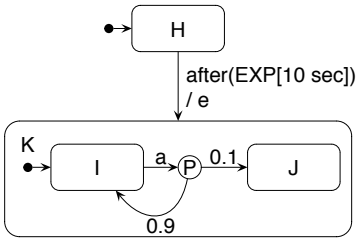

Figure 2. Example StoChart

StoChart *semantic model*. The formal semantics of StoCharts [26] is defined in terms of an extension of labelled transition systems, called *stochastic I/O-automata* (Iosa, for short). Basically, an Iosa is an automaton with *locations* representing the possible states of the system, and transitions between locations representing the system's evolvement. These transition systems are equipped with *timers* to model stochastic delays, and with a set of *actions* to model system activities. The use of timers in transition systems is very much in line with the use of clocks in, e. g., timed automata [4, 29]. While clocks in timed automata run forward at the same pace and are always reset to 0, our timers are initialised by sampling a stochastic distribution and run backwards, all at the same pace. On the other hand, our timers are always checked for expiration (i. e., is the timer equal to zero?), while clocks can be checked against complex conditions.

Input and output actions are distinguished to allow for the composition of transition systems, like in I/O-automata [30]. Three types of transition relations are used: input transitions, output transitions, and delay transitions, the latter being enabled once a timer expires. Whereas input and delay transitions are standard ternary relations (input is even functional), the output transition relation is probabilistic. Like in I/O-automata we assume input-enabledness, i. e., in each location any input must be accepted.

Figure 3 shows an example Iosa. It corresponds to the parallel composition of the statechart diagram in figure 1 (which is a trivial StoChart) with the StoChart in figure 2. To simplify the figure, we have assumed that the system is *closed*, i. e., no further external inputs reach the system. Each state of the Iosa is drawn as a box which contains the active StoChart nodes and the events that have been received but not yet processed. The after construct is translated to setting a timer and waiting until it expires: in the initial state, the timer is set according to an exponential distribution with a mean of 10 seconds. After expiration of the timer, the event e is generated by the StoChart. This triggers the statechart diagram, as described above, to generate events a and f. Now, event f triggers the UML statechart diagram to leave *and*-node G entering node B, while

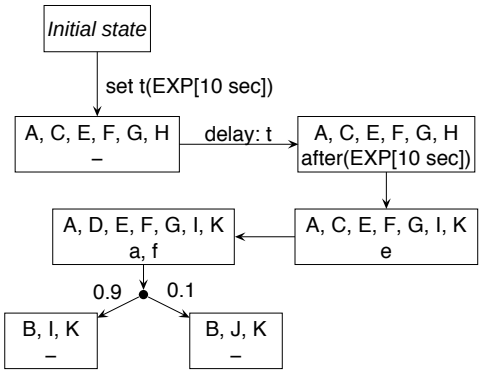

Figure 3. Example Iosa

event a simultaneously triggers the probabilistic choice between node I and J in the StoChart. The other probability spaces for output transitions are trivial: they assign probability 1 to a single next state.

Relation to Generalised Semi-Markov Processes (GSMP). A Iosa is a specific semantic structure that contains exactly the ingredients needed for StoChart semantics [26]. GSMPs are a frequently found model for stochastic processes, and often, a Iosa can be translated to a GSMP. However, GSMPs do not allow nondeterministic choice, and they also restrict the allowed stochastic distributions so as to reduce the probability that two timers expire at the same moment (introducing nondeterminism again) to zero. The Iosa associated with the StoChart modelling our case study does not contain any nondeterminism and therefore allows GSMP-based analysis.

3. ETCS radio connection dependability

This section presents the high-speed train radio signalling case study we consider, which is inspired by work of Zimmermann and Hommel [40].

European Train Control System. The upcoming European Train Control Systems (ETCS) serves as a unifying standard of many European railways. It is promoted by the European Union to simplify access to and cross-border traffic between different national rail networks. The main constituent is a uniform communication infrastructure across Europe. This communication infrastructure is based on GSM-R, which is an adaptation of the well-known GSM protocol for wireless communications to railway specific applications. For example, while shunting (moving wagons to compose new trains), often a couple of wagons are pushed and a shunting worker needs to guide the train driver in real-time. Tailored to this case, GSM-R specifies a "link assurance signal" which confirms to the driver that the voice communication link exists.

The specifications of GSM-R and of ETCS contain several quality-of-service requirements. In the case study we consider, we focus on the reliability of the communication between the train and the trackside control authority and assess the impact communication delays may have on safety distances between consecutive trains.

ETCS levels. ETCS knows multiple *levels* to enable gradual migration from the current systems. At all levels, so-called *Eurobalises* are used. A Eurobalise is a transponder that sends a message to a train passing over it. At level 1, this is the main communication medium; among others, Eurobalises are used to ensure mutually exclusive access to track segments. They transmit so called *movement authorities* and other information on danger points to the train, similar to current systems that warn the train driver when approaching a danger point. A movement authority grants the train exclusive access to a track segment ahead of the train, called a track *block*. At levels 2 and 3, GSM-R-based radio communication is used for transmitting both track-specific information as well as movement authorities, while Eurobalises are only used for exact position measurement. At these levels, the dispatching centre that assigns movement authorities to trains is called *radio block centre* (RBC). Levels 2 and 3 differ by how the integrity of the train is checked. At level 2, traditional track-side axle counters or rail circuits are in use, being installed along the track at regular intervals. They form boundaries of fixed blocks, which at any time can only be used by one particular train, namely the one which possesses the movement authority for the block. Level 3 trains instead are equipped with an on-board integrity checking device. This enables the system to declare the track behind the train clear with virtually no delay, which in turn is a requirement for so-called *moving-block* operation, where the track block granted exclusively to a specific train is not a fixed unit of the track between two axle counters, but instead moves with the train along the track.

Headway. This moving-block operation is expected to reduce the *headway,* i. e., the time between the passage of consecutive trains at some point of the track, well below 3 minutes, which is the usual headway in fixed block operation. The minimal headway is the sum of several delays (in this calculation, we assume trains running at 300 km/h): a delay needed for train integrity check ($<$ 4 seconds), the communication delay itself, and a delay that reflects certain physical distances: *(i)* train length (typical value: 400 m), *(ii)* braking distance (about 2500 m), *(iii)* margin for position measurement errors (5 %). The latter is at most 50 m, if Eurobalises are positioned no more than 1 km apart. For simplicity, we assume that these distances sum up to 3000 m. The train travels this distance in 36 seconds. Thus, with continuous instant communication, 40 seconds would be the ultimate lower bound on the headway between consecutive trains.

In this case study, we will have a closer look at the reliability of communication needed for moving-block operation. GSM-R may fail to establish a connection, a connection may get degraded or lost; messages may get delayed during handover from one GSM radio cell to another. Under normal circumstances, the train reports the safe position of its head and tail at fixed intervals, for example every 5 seconds. What happens if one or several of these reports get lost? On the other hand, movement authorities need to be transmitted to the train at similar intervals. What is the probability that the train misses a movement authority?

Assumptions. We assume that the GSM-R network functions as specified in the Euroradio specification [18, 40]. In particular, we assume:

- The delay to establish a GSM-R connection is at most 5 seconds with 95 % and at most 7.5 seconds with 99.9 % probability. Delays of more than 7.5 seconds are regarded as connection establishment errors.
- The end-to-end delay of a (short) message is at most 0.5 sec with 95 %, at most 1.2 sec with 99 % and at most 2.4 sec with 99.99 % probability.
- Handover takes place whenever the train passes from one GSM radio cell to another. As ETCS is intended to work with train speeds up to 500 km/h, we take a pessimistic assumption on the time between handovers. The mean distance between handovers is specified to be 7 km; this leads to a mean time between cell handovers of 50 seconds. The communication break during handover lasts at most 0.3 sec.
- From time to time, the train may pass an area where communication is degraded and frequent transmission errors occur. These periods are more than 7 seconds apart with 95 % probability. A degraded period is required to be shorter than 1 second with 95 % probability.
- A connection loss has a probability $< 10^{-4}$ per hour. It shall be detected within 1 sec.

With respect to the train-specific behaviour, we adhere to the following assumptions as put forward in [19, 34]:

- A passenger train completes an integrity check within 4 seconds; it reports the outcome and its position to the RBC at most once in 5 seconds.
- A typical train trip has a duration of 1 hour.

Our Focus. We view all the above properties as constraints to be met by the environment in which a level 3 train operates. This view can be seen as an application of the *design-by-contract* paradigm [32, 7], in the sense that the ETCS system is required to work properly if these constraints are

met (or outbalanced). The question then remains what specific guarantees can be distilled from these assumptions. We intend to check whether it is possible that trains run at 300 km/h with only a small headway (about 1 minute). In particular, we want to find answers to the following questions:

- The probability p that a message is transmitted successfully has to be at least 99.95 %. This figure is based on the availability requirement of [18]. As parts of the communication delay are distributed stochastically, the success probability depends on the time frame t we allow as maximal communication delay. Recall that the minimal headway with (hypothetic) instant communication is 40 seconds. With a 1 minute headway in mind, the question is: Is the probability $p > 99.95\,\%$ for $t = 20$?
- Even if 20 seconds lead to p being in the range required above ($\geq 99.95\,\%$), it is still not obvious that this also enables trains to run at a headway of about 1 minute for a full hour. We therefore also consider the question: What is the probability that two trains run for a complete trip with a small headway without ever braking or stopping?

4. StoChart model

To assess the questions raised above, we have developed a STOCHARTS model of the ETCS case. The model has been built in a systematic and incremental way, starting from the most basic operation, which represents a simplified view on the normal operation. After that, we added processing of failures one by one. In our subsystem, there are two relevant components: the train and the radio block centre.

Figure 4 shows the train model. In node Reporting position, a position report is prepared every 5 seconds, indicated by the after edge with a deterministic delay. It is sent as soon as possible, if the train assumes there is a connection to the RBC. The node Connection status just stores the information the train has about the connection: it is either disconnected (the initial node, where it tries to establish a connection by sending event try), connected normally or involved in a cell handover.

There are several causes for delay and stochasticity in the communication protocol between train and RBC. We have decided to incorporate these delays into the model of the RBC, while the train model is reactive, waiting for stimuli from the RBC. For example, to model the establishment phase for a radio connection, the RBC model includes after(...) operators and sends a message to the train system when the connection is established. Alternatively, we could have split the communication characteristics from the RBC channel management, and let the train and RBC interact through this third submodel.

Figure 5 shows the RBC model together with the delays. When the train tries to establish a connection with the RBC (by sending event try), the connection establishment delay starts. It is guaranteed to be at most 5 seconds with 95 % probability and at most 7.5 seconds with another 4.9 %. We have modelled this guarantee using two deterministic delays, one of length 5 sec (on the edge leaving node connecting 1), the other one of length 2.5; Alternatively, we could have modelled it using a single more complex distribution.

Actually, the node connected contains a subchart to model the communication delay (0.5 seconds with 95 % probability, 1.2 with 99 % probability, 2.4 with 99.99 % probability). This subchart is similar to the model for the connection establishment delay. In order not to clutter the pictorial representation of the STOCHART this subchart is omitted from figure 5, but it is part of the model we study.

This basic model of the RBC reflects normal operation. To this, we have added three relevant possibilities of perturbations. During the design of our model, they have been added one after another: *(i)* During *cell handover,* the connection is lost for at most 0.3 sec. As argued above, the connection between RBC and (moving) train is subject to a cell handover about once in 50 seconds. This is modelled in the subnodes of node correct. The cell handover is visible for the train (via events ch_st and ch_en). *(ii)* Due to signal obstruction, transmission errors are common in wireless communication, but error correcting codes make it possible to correct certain limited amounts of bit-errors. However periods of *frequent* transmission errors may occur, making it impossible to recover sent messages from the received bitstream. In our model, this is reflected by node error burst. Both the beginning and the end of the error burst period are modelled by exponential delays, as the errors occur stochastically. The mean times of the relevant delays are chosen as to meet the requirements given above. The node error burst may be entered from either node in the *or*-node correct. In particular, we assume that if an error burst occurs during cell handover, the latter is aborted, and node connected is reentered once the burst is over. As another possibility, we could have modelled that the error burst maintains the node identity inside correct. We discuss this alternative in Section 6. *(iii)* All other failure types are subsumed under *connection loss,* which is required to happen at most 10^{-4} times per hour. We have modelled this by an exponential delay with an average of 10^4 hours. The train notices the connection loss with a delay; this is modelled by waiting in node undetected connection loss for 1 second.

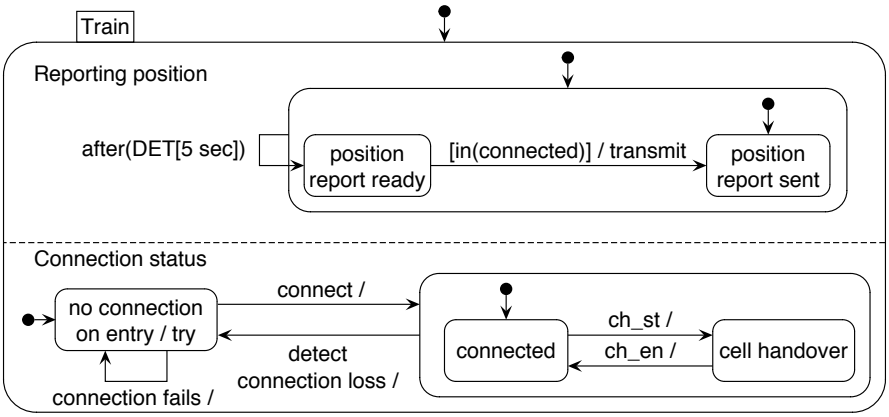

Figure 4. The train model

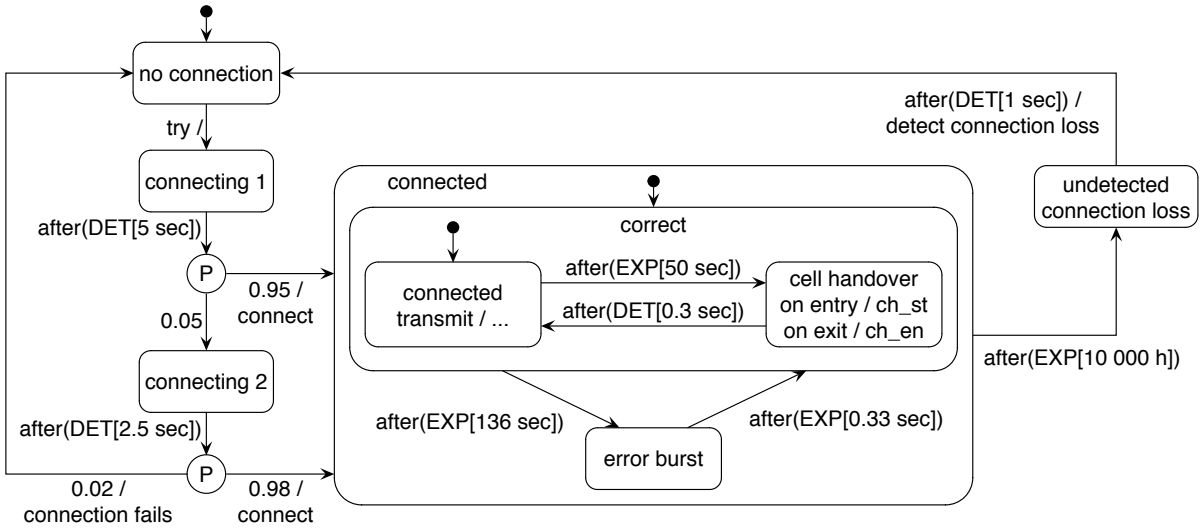

Figure 5. The radio block centre model, including error models

5. Analysis of the example

Model analysis of a collection of STOCHARTS is based on the analysis of the associated IOSA. This model can be analysed using simulation, or using model-checking. In the most general case model-checking algorithms in the style of [13] are needed. If nondeterminism is absent (which is the case in our example), one can deal with algorithms for GSMPs, such as the ones implemented in PROVER [39]. This tool uses *discrete event simulation* to estimate probabilities of interesting system behaviours, similar to various other tools for GSMP analysis. PROVER is, however, particular in the manner the behaviour-of-interest is specified by the user. It allows to specify two types of *requirements*. First, PROVER can estimate the probability with which certain path-based system requirements are satisfied by the model. These requirements are stated as *path* properties in the stochastic temporal logic CSL [5]. Second, PROVER can also be used similar to a model checker to verify whether a CSL *state* property is met by the model. So, the main differences to other simulation tools lies in the fact that requirements are specified as CSL properties, and that it can be used to verify (or refute these requirements). We have chosen PROVER because this type of verification fits well in the *design-by-contract* paradigm we follow, providing (probabilistic) guarantees on the design under consideration. This choice, however, is not in the heart of the STOCHART approach, we could equally well have been taken another more standard discrete-event simulation package.

Experiments. The IOSA models associated with the STOCHART specifications were generated in a semi-automatic way, and manually translated into the input format of PRO-

VER. All experiments were performed on a Dell Latitude D600 with 1 GB memory and a processor speed of 1.7 GHz. We have analysed the following two variants of train communication:

- A simple version of the system, with a single train and a communication link as shown in the STOCHARTS. In this setting, we assume that the train generates a position report every 5 seconds and sends it off to the RBC once it assumes the connection is working. We have checked whether the communication is reliable enough, depending on the delay between the reception of successive position reports. Recall that the probability that a message is received is required to be at least 99.95 %. Stated as a CSL property, the requirement is:

$$\mathcal{P}_{\geq 0.9995}(\diamond^{\leq t}\text{message received})$$

We should check this property in the state where a position report has just been generated, sent and received without delay (an over-approximation of the best-case behaviour for the preceding message). PROVER has checked the property (in what we have called the model-checker like mode above) for $t = 20$ seconds (5 seconds delay between successive position reports *plus* 15 seconds communication delay) and produced the result: the success probability p is ≥ 0.9995 with a confidence > 0.9999. PROVER runtime was 21 seconds.

In addition, we used the tool to estimate the probability p that the communication succeeds within at most t seconds. The table below gives the estimates.

t	p	error estimate	runtime
10	0.98267	$\pm 9 \cdot 10^{-5}$	1 min 36 s
15	0.999700	$\pm 9 \cdot 10^{-6}$	2 min 16 s
20	0.9999944	$\pm 6 \cdot 10^{-7}$	14 min 18 s
25	0.99999995	$\pm 5 \cdot 10^{-8}$	26 min 53 s

We can see that even for $t = 15$ seconds, the success probability would have been large enough. Checking the above CSL property again with this parameter value shows that also for $t = 15$ seconds, the success probability p is ≥ 0.9995 with a confidence > 0.9999. However, PROVER needed 55 seconds; this indicates that more samples had to be generated to verify the property.

- A model for two successive trains running at 300 km/h. In this setting, we assume that the train in front sends a position and integrity report to the RBC, which in turn sends a movement authority to the following train. The delay between two receptions of a movement authority is required to be not too long, to avoid that the following train needs to brake. On the other hand, as the above experiments have shown, this delay can be more than 10 seconds with a non-neglectable probability. The model for the second radio link (from RBC to the following train) is the same as for the first one; therefore, we omit the STOCHARTS for this link.

With PROVER, it is difficult to measure the age of the information on which the movement authority received is based. Therefore, we measure the time Δt between two successive receptions of movement authorities. A simple calculation shows that then, the second movement authority is based on information not older than Δt + 7.4 seconds. With a headway of slightly above one minute, namely 62.4 seconds, a maximum Δt of 15 seconds is fine. Otherwise, the following train will have to brake at least once during the trip. PROVER provides us with the following results:

max Δt	p	error estimate	runtime
> 10	0.9562	$\pm 9 \cdot 10^{-4}$	17 min 34 s
> 15	0.101	$\pm 2 \cdot 10^{-3}$	27 min 5 s
> 20	0.0036	$\pm 4 \cdot 10^{-4}$	35 min 55 s
> 25	0.00034	$\pm 11 \cdot 10^{-5}$	39 min 28 s

We can see that about one train out of 9 or 10 would have to brake because of a communication failure. However, in most cases, the failure is short, and the train could recover after a very short braking period. We can conclude that the ETCS radio link is reliable enough to run trains with a headway of just above 1 minute.

If trains are scheduled at a larger headway, one might try and relax some of the QoS assumptions on the GSM-R network. For example, telecom operators typically assume a higher connection loss probability than 10^{-4} per hour for traditional GSM networks. Relaxing these QoS assumptions may allow for a cheaper GSM-R infrastructure.

6. Discussion

This section provides a discussion of our experiences when modelling and analysing the ETCS case with STOCHARTS and PROVER.

STOCHARTS *and ETCS.* We have given a detailed description of our efforts to model the a safety critical application with a UML-based stochastic modelling language. In a nutshell, we have studied the circumstances under which high-speed trains may follow each other with a headway of about 1 minute while driving with 300 km/h, without running a substantial risk that wireless communication faults may cause them to brake. This observation may lead to a better utilisation of European railtracks as traditional headways are about 3 minutes.

Our results seemingly contradict what is reported in [40], where the probability that a train may need to stop because

of a communication failure may be as high as 94 %. We mention however that we use an approximately three times longer value for the additional headway, and use a more faithful model of communication delays. Therefore, the two analysis results are uncomparable.

History connector of UML statechart diagrams. On the modelling side, some of our choices eased modelling with the basic concepts of UML statechart diagrams. In particular, we have assumed that if an error burst occurs during cell handover, the latter is aborted, and the initial node is reentered once the burst is over. As another possibility, we could have modelled that the error burst maintains the node identity inside the embracing *or*-node. One of the possibilities to achieve this with UML notation is provided by a history connector (denoted Ⓗ) which indicates that the substatechart should memorise which of its child nodes was active last and reenter that child node (instead of the initial one) in case the *or*-node is reentered. This however raises the question how to handle the time that has elapsed in this node prior to leaving it. In our stochastic timed interpretation, this opens a field of interpretation which closely resembles the *firing* or *execution policy* issue in non-Markov stochastic Petri nets [1, 8, 15] and activity networks. We indeed believe that the history connector needs to be considered parametric in this policy.

Tool support. The design of STOCHART is supported by the general editing tool TCM [16]. However, when it comes to the derivation of the underlying IOSA, and then to the resulting GSMP, several steps are still done manually. This is unsatisfactory, because this manual process is both cumbersome and error-prone. We are exploring ways to effectively implement the operational semantics of STOCHART. Apart from a direct implementation along the lines of [26, 24], we are also considering a translational semantics, which maps a collection of STOCHARTS to the MODEST language [12]. Such a link enables mechanisation of the stochastic analysis of STOCHARTS, because MODEST is connected to the discrete event simulation engine of the MÖBIUS toolset [14].

Acknowledgement. We thank Yaroslav Usenko from Universiteit Twente, who helped us in finding an error in a preliminary version of the models.

References

[1] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. The effect of execution policies on the semantics and analysis of stochastic Petri nets. *IEEE Transactions on software engineering*, 15:832–846, 1989.

[2] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modeling with generalised stochastic Petri nets*. Wiley, 1995.

[3] M. Ajmone Marsan, G. Conte, and G. Balbo. A class of generalised stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on computer systems*, 2(2):93–122, 1984.

[4] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[5] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.

[6] S. Bernardi, S. Donatelli, and J. Merseguer. From UML sequence diagrams and statecharts to analysable Petri net models. In S. Balsamo, P. Inverardi, and B. Selic, editors, *Proceedings of the third international workshop on software and performance*, pages 35–45, New York, 2002. ACM Press.

[7] A. Beugnard, J.-M. Jézéquel, and N. Plouzeau. Making components contract aware. *Computer*, 32(7):38–45, 1999.

[8] A. Bobbio, A. Puliafito, and M. Telek. A modeling framework to impleemnt preemption policies in non-Markovian SPNs. *IEEE Transactions on software engineering*, 26(1):36–54, 2000.

[9] C. Cavenet, S. Gilmore, J. Hillston, and P. Stevens. Performance modelling with UML and stochastic process algebra. In *UK performance engineering workshop (UKPEW)*, page 16, 2002.

[10] C. Constantinescu. Impact of deep submicron technology on dependability of VLSI circuits. In *International conference on dependable systems and networks (DSN'02)*, pages 205–209. IEEE, 2002.

[11] W. Damm, B. Josko, H. Hungar, and A. Pnueli. A compositional real-time semantics of STATEMATE designs. In W.-P. de Roever, H. Langmaack, and A. Pnueli, editors, *Compositionality, the significant difference : intl. symposium COMPOS '97*, volume 1536 of *LNCS*, pages 186–238, Berlin, 1997. Springer.

[12] P. R. D'Argenio, H. Hermanns, J.-P. Katoen, and R. Klaren. MoDeST : a modelling and description language for stochastic timed systems. In L. de Alfaro and S. Gilmore, editors, *Process Algebra and Probabilistic Methods. Performance Modelling and Verification: Joint International Workshop, PAPM-PROBMIV*, volume 2165 of *LNCS*, pages 87–104, Berlin, 2001. Springer.

[13] L. de Alfaro. How to specify and verify the long-run average behavior of probabilistic systems. In *Thirteenth annual IEEE symposium on logic in computer science*, pages 454–465, Los Alamitos, Calif., 1998. IEEE computer society.

[14] D. D. Deavours and W. H. Sanders. Möbius : Framework and atomic models. In *Proc. 9th Int. Workshop on Petri Nets and Performance Models (PNPM '01)*, pages 251–260. IEEE, 2001.

[15] D. D. Deavours and W. H. Sanders. The Möbius execution policy. In *Proceedings of the 9th intl. workshop on Petri nets and performance models*, pages 135–144. IEEE, 2001.

[16] F. Dehne, H. van de Zandschulp, and R. Wieringa. Toolkit for conceptual modeling (TCM). `http://www.cs.utwente.nl/~tcm/`.

[17] R. Eshuis and R. Wieringa. Requirements-level semantics for UML statecharts. In S. F. Smith and C. L. Talcott, editors, *Formal Methods for Open Object-Based Distributed Systems IV : ... FMOODS*, pages 121–140, Boston, 2000. Kluwer Academic Publishers.

[18] Euroradio FFFIS : class 1 requirements. `http://www.aeif.org/db/docs/ccm/SUBSET-052_v200.PDF`, 2000.

[19] Functional requirement specifications : train integrity monitoring system. `http://www.aeif.org/db/docs/ccm/EEIG-TIMS-Document.doc`, 2000.

[20] N. Götz, U. Herzog, and M. Rettelbach. *Multiprocessor and distributed system design : the integration of functional specification and performance analysis using stochastic process algebras*, volume 729 of *LNCS*, pages 121–146. 1993.

[21] D. Harel and A. Naamad. The STATEMATE semantics of statecharts. *ACM Transactions on Software Engineering and Methodology*, 5(4):293–333, 1996.

[22] J. Hillston. *A Compositional Approach to Performance Modelling*. Distinguished dissertations in computer science. Cambridge University Press, Cambridge, 1996.

[23] G. Huszerl, I. Majzik, A. Pataricza, K. Kosmidis, and M. Dal Cin. Quantitative analysis of UML statechart models of dependable systems. *The computer journal*, 45(3):260–277, 2002. ISSN: 0010-4620.

[24] D. N. Jansen. *Extensions of statecharts with probability, time, and stochastic timing*. PhD thesis, Universiteit Twente, Inmarks, Bern, October 2003. ISBN 3–9522850–0–5.

[25] D. N. Jansen, H. Hermanns, and J.-P. Katoen. A probabilistic extension of UML statecharts : specification and verification. In W. Damm and E.-R. Olderog, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems : ... FTRTFT*, volume 2469 of *LNCS*, pages 355–374, Berlin, 2002. Springer.

[26] D. N. Jansen, H. Hermanns, and J.-P. Katoen. A QoS-oriented extension of UML statecharts. In P. Stevens, J. Whittle, and G. Booch, editors, *«UML» 2003 : the unified modeling language*, volume 2863 of *LNCS*, pages 76–91. Springer, 2003.

[27] P. King and R. Pooley. Derivation of Petri net performance models from UML specifications of communications software. In B. R. Haverkort, H. C. Bohnenkamp, and C. U. Smith, editors, *Computer Performance Evaluation, Modelling Techniques and Tools : 11th intl. conference, TOOLS*, volume 1786 of *LNCS*, pages 262–276, Berlin, 2000. Springer.

[28] C. Lindemann, A. Thümmler, A. Klemm, M. Lohmann, and O. P. Waldhorst. Performance analysis of time-enhanced UML diagrams based on stochastic processes. In S. Balsamo, P. Inverardi, and B. Selic, editors, *Proceedings of the third international workshop on Software and performance*, pages 25–34, New York, 2002. ACM Press.

[29] N. Lynch and F. Vaandrager. Forward and backward simulations : II. timing-based systems. *Information and Computation*, 128:1–25, 1996.

[30] N. A. Lynch and M. R. Tuttle. An introduction to input/output automata. *CWI quarterly*, pages 219–246, 1989.

[31] J. Merseguer, S. Bernardi, J. Campos, and S. Donatelli. A compositional semantics for UML state machines aimed at performance evaluation. In M. Silva, A. Giua, and J. M. Colom, editors, *Proceedings of the 6th international workshop on discrete event systems*, pages 295–302. IEEE computer society press, October 2002.

[32] B. Meyer. Applying "design by contract". *Computer*, 25(10):40–51, October 1992.

[33] P. Mitton and R. Holton. PEPA performability modelling using UML statecharts. In N. Thomas and J. Bradley, editors, *UKPEW 2000 : Sixteenth Annual UK Performance Engineering Workshop*, 15 p. University of Durham, UK, July 2000. `http://www.dur.ac.uk/nigel.thomas/UKPEW2000/online-proceedings.html`.

[34] Performance requirements for interoperability. `http://www.aeif.org/db/docs/ccm/SUBSET-041_v200.PDF`, March 2000.

[35] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

[36] D. Tennenhouse. Proactive computing. *Comm. of the ACM*, 43(5):43–50, 2000.

[37] *UML profile for modeling quality of service and fault tolerance characteristics and mechanisms : request for proposal*. Object Management Group, Inc., Needham, MA, January 2002. `http://www.omg.org/cgi-bin/apps/do_doc?ad/02-01-07.pdf`.

[38] *UML Profile for Schedulability, Performance, and Time Specification : OMG Adopted Specification*. Object Management Group, Inc., Needham, MA, November 2002. `http://www.omg.org/cgi-bin/doc?ptc/2002-03-02`.

[39] H. L. S. Younes and R. G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In E. Brinksma and K. G. Larsen, editors, *Computer aided verification : 14th intl. conference, CAV*, volume 2404 of *LNCS*, pages 223–235, Berlin, 2002. Springer.

[40] A. Zimmermann and G. Hommel. A train control system case study in model-based real time system design. In *International parallel and distributed processing symposium*, page 118b. IEEE, 2003.