Analysis of Safety-Critical Cloud Architectures with Multi-Trajectory Simulation

Armin Zimmermann, Technische Universität Ilmenau

Thomas Hotz, Technische Universität Ilmenau

Volker Hädicke, Thales Transportation Systems Deutschland

Martin Friebe, Thales Transportation Systems Deutschland

Key Words: Discrete Event Modeling & Simulation, Fault Tolerance and Safety Critical Systems, Reliability Modeling

SUMMARY

Dynamic safety-critical systems require model-based techniques and tools for their systems design. The paper presents a stochastic Petri net model of an industrial safety-critical cloud server architecture for train control. Its reliability has to be evaluated to assess tradeoffs in architecture and level of fault tolerance. Simulation methods are too slow for such rare-event problems, while numerical analysis techniques suffer from the state-space explosion problem. The paper extends a recently developed multi-trajectory simulation algorithm combining elements of simulation and numerical analysis such that it increases the accuracy of rare-event simulations within a given computation time budget. Simulation experiments have been carried out with a prototype tool.

1 INTRODUCTION

The systematic design of complex technical systems requires prediction of non-functional properties such as reliability of a planned architecture in many fields of engineering applications [1]. For safety-critical systems and highly reliable designs, such properties are of equal importance as the functional requirements, especially when a certification is necessary before use. This is for instance mandatory in transportation systems including train control or avionics.

The paper presents a model of a safety-critical cloud architecture composed of hardware elements and software processes extending the work of [2]. The application example is derived from an industrial project towards a future cloud-based solution for a safety-critical train control system. Such systems introduce design flexibility and have the advantage of allowing dynamic fault tolerance methods, which may achieve reliability, availability, maintainability, and safety (RAM) levels with less hardware than a simple static parallel architecture. A good design solution will fulfill the necessary RAM requirements in a certifiable way with a software and hardware solution that is efficient in terms of investment and running cost.

Such design decisions should be based on the analysis of design variants and parameters. Moreover, a time-efficient method to derive RAM values from models as well as a software tool to create and evaluate them are necessary. There are many model types available in the literature for reliability evaluation [1, 3, 4]; their industrial use is often restricted to static models including fault trees (FT) and reliability block diagrams (RBD). The reliability of complex technical systems is, however, often achieved by fault-tolerance and reconfiguration, or in general depends on the dynamic behavior [3, 5]. This is certainly true for the type of systems considered here, and thus static FTs and RBDs cannot be used.

Common models for discrete-state (discrete-event) systems performance and RAM evaluation include Markov chains, queueing networks, and stochastic Petri nets [1]. System designers may also use simulation tools that require a system specification in a programming-language like style, but while that allows a great expressiveness, it does not have the advantages of graphical models for communication and numerical analysis.

The safety-critical cloud architectures covered in this paper are a typical example of dynamic systems with concurrent processes, shared resources, and synchronization patterns. A natural choice to model such systems are stochastic Petri nets [6, 7], which are well established for dynamic, distributed systems and their reliability evaluation [5, 8]. Their industrial use is supported by the norm IEC 62551 [9]. Simpler dynamic models with memory-less stochastic behavior can be analyzed with Markov chains [3, 10].

The two main methods for the performance/reliability evaluation of such models are numerical analysis and simulation [1, 6]. Numerical analysis requires generating and analyzing the full reachability graph of all possible states, which may be too large to be handled computationally. Moreover, the analysis gets hard or intractable when there are non-memoryless delay distributions present [11]. The results, however, are exact apart from possible numeric rounding errors. The computation time does not suffer from the typical mix of fast and slow activities in reliability models, but depends non-linearly on the number of states and state transitions.

Models with a large state space or non-Markovian delay distributions require simulation as the second option for their evaluation [12]. This is often the case for technical systems with their inherent clocking, deadlines, deterministic schedules, and Weibull-estimated life times of modules. An open problem is then, however, the very long run time of simulations to collect enough significant samples for statistically sound estimation of low probabilities for highly reliable systems. This problem is known in the literature as rare-event simulation, and possible techniques to tackle it are importance sampling [13] and splitting [14, 19]. Both strategies enforce certain trajectories in the simulation, to get more samples of interest with the same amount of simulated events. The results have to be transformed to consider the changes. The main issue with generalizing these methods is the configuration effort to achieve a good speedup.

Thus, simulation and numerical analysis have their advantages for different problem types. The recently developed *multi-trajectory simulation algorithm* [15] combines elements of both in a hybrid fashion. The main idea is to store several simulation trajectories (or particles) and progress them in parallel, while collecting the trajectories that merge. The main configuration parameter of the algorithm is the number of particles to be stored: if it is set to one, the algorithm works exactly like a regular simulation. If it is set to a number high enough to store the full reachability set, the method works similar to the transient numerical evaluation of a discrete-time Markov process that converges to the desired stationary solution rapidly. The method is introduced and its unbiasedness is proved in [15]. The example application models presented therein showed the principal advantages, as the algorithm covers models that should be evaluated by either simulation or numerical analysis automatically in an efficient way.

The application to reliability problems with their rare-event behavior has not been considered so far, which is a contribution of this paper. The paper extends the method to speed up reliability evaluation of systems considerably compared to simulation. The open research issue is then how the particle-controlling heuristic decisions of the multi-trajectory simulation algorithm should be configured for a model. We show how such a heuristic can be derived for the example Petri net model based on approaches in splitting simulations [16, 19]. A prototypical implementation in the tool TimeNET [17] is presented as well, which is used to compute numerical results and run times to check the computational efficiency.

The contribution of the paper includes 1) a template stochastic Petri net model for safety-critical cloud architectures in a transportation system use case; 2) the first application of the multi-trajectory simulation algorithm to an industrial reliability problem; 3) a novel heuristic to control the trajectory management of the algorithm for the application model, and 4) its prototype implementation in a software tool.

2 DYNAMIC RELIABILITY MODELS FOR SAFETY-CRITICAL CLOUD ARCHITECTURES

This section describes the type of application architectures considered in this paper, and introduces a stochastic Petri net model for an industrial case study.

2.1 Safety-Critical Cloud Architectures

A cloud is an environment that provides on-demand availability of computer system resources for programs and applications to operate on. Resources are processing power, memory, the file system and networking. A cloud consists of an arbitrary number of server clusters. Model-based systems engineering of cloud server systems has been studied in the literature, covering for instance power consumption and reliability [18] or disaster-tolerance by evaluating geographical locations [2].

The basic template architecture followed in this paper is shown in Figure 1: Horizontally, the lowest level of a cloud system is based on physical machines, which are processing each their own operating system. Each physical machine together with its own operating system is then merged into one (Infrastructure as a Service) IaaS layer. At the next level, the IaaS layer is partitioned, which provides Platforms as a Service (PaaS) for the Virtual Machines (VMs) to operate on. Each VM can freely move within the cloud and thus provide the operational environment for applications.

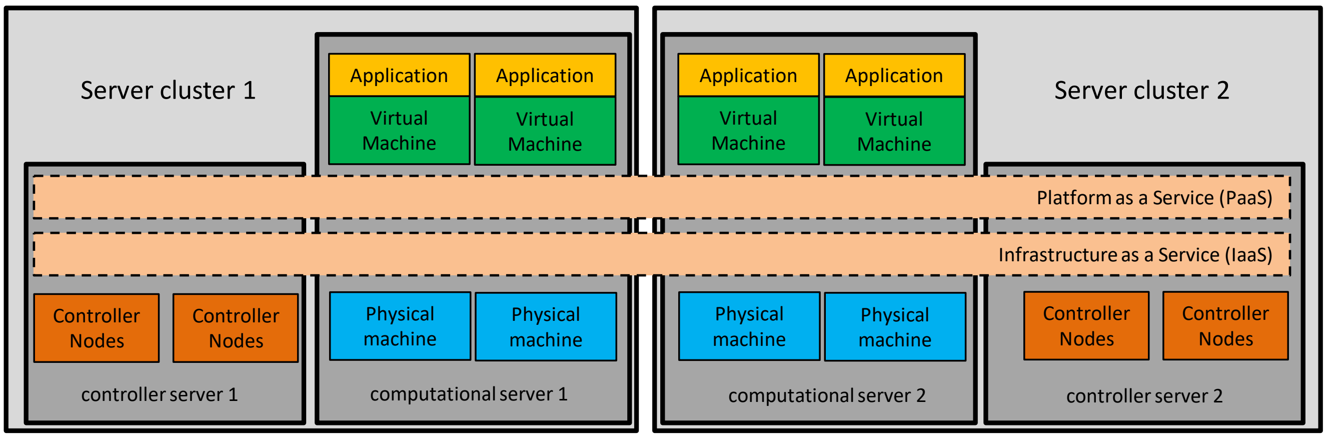

Figure 1 – Sample architecture of a safety-critical cloud architecture

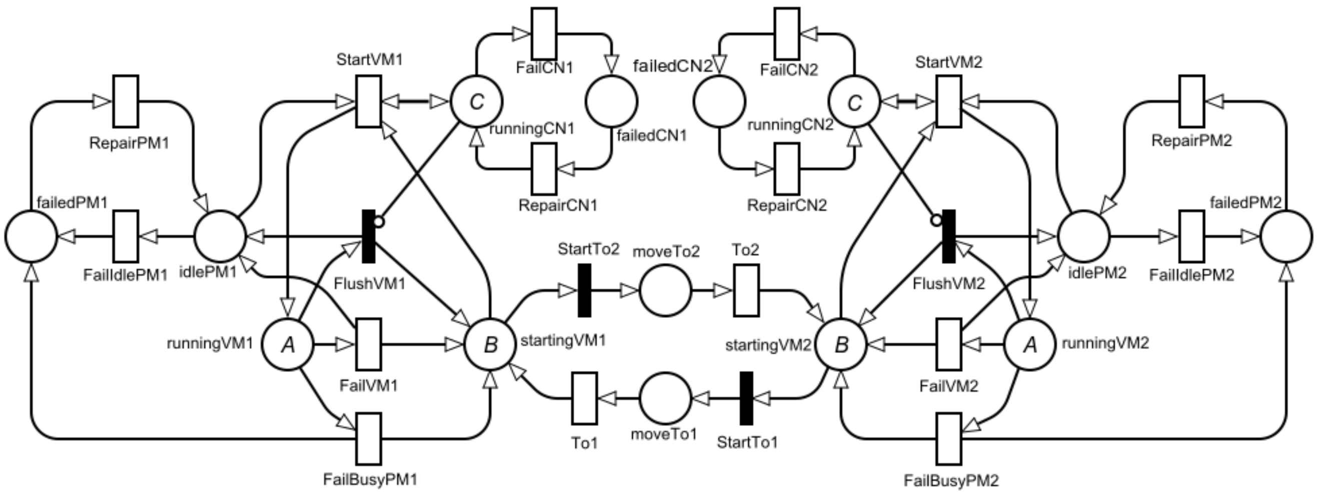

Figure 2 – Stochastic Petri Net for the Example Setup

Vertically, a cloud can be separated into an arbitrary number of server clusters, where each server cluster consists of controller servers and computational servers. Computational servers provide the actual computational resources, whereas controller servers manage and operate the available resources. In the present model, there are two server clusters with computational and controller server 1 and computational and controller server 2. Both server clusters are geographically separated and thus connected through a network link.

The architecture and model presented in this paper have been developed for an industrial project in the field of train control systems engineering. The very strict safety requirements in this field can be achieved with the type of fault tolerance that the considered cloud architecture offers, when it is configured with dedicated application-specific resources. The fitness of the design must be proven during the certification process, for which a formal model and evaluation technique are needed.

2.2 Stochastic Petri Net Models for Safety-Critical Clouds

Figure 2 introduces a stochastic Petri net model for the described safety-critical cloud system with two server clusters containing A physical machines and C controller nodes each, which should run a set of virtual machines with applications. It shows the general modular construction of such models for the description of dynamic failure, repair and relocation activities. The model construction extends the work of Silva et.al. [2] for the dedicated safety-critical architecture considered in this paper.

Each server cluster consists of a certain number of Physical Machines (PM), Virtual Machines (VM) and Controller Nodes (CN). The purpose of the PMs is to provide services such as computational power and storage for the VMs, whereas the VMs represent the actual Application operating on the PM. The CNs' purpose is to enable the individual PMs to function as one unified layer and as a necessary requirement for the VMs to operate.

Each server has a certain number of operating VMs (runningVM1), which can fail either due to either a physical failure (FailBusyPM1), a software failure (FailVM1) or due to the loss of CNs (FlushVM1). In case of the physical failure, the PM goes into a failed state PM1, where it gets repaired (RepairPM1) and eventually awaits the next server request as an idle PM (idlePM1). In the second case, a running VM can fail due to software failure, after which the VM reboots (startingVM1) and the PM immediately becomes available (idlePM1) for the next server request. Other than that, a running VM stops operating when all CNs of the cluster stop their operation. The model in the figure shows the initial Petri net marking in the regular running state, where A VMs are running on a PM each (thus there are no additional idle PMs, and all PMs are active). Further waiting VMs are in places startingPM, thus the overall number of virtual machines in the cloud system is $2(A+B)$.

The restarting process of a VM requires three sub-processes to be successful: A VM in its booting process (startingVM1) can only boot and run (runningVM1), when CNs are operational (runningCN1) and PMs are available (idlePM1). Failed PMs (failedPM1) have to get repaired (RepairPM1) to go into an operational, but idle state (idlePM1), from where they could eventually fail again (FailIdlePM1). Similarly to the PMs, CNs (runningCN1) also can fail (FailCN1) from where they have to get repaired (RepairCN1) before resuming their function (runningCN1).

The modeling assumptions are as follows:

- hardware (HW) and software (SW) failure times are exponentially distributed
- HW mean time to failure for PMs and CNs is identical at 4000 hours

- HW mean time to repair is 12h
- SW mean time to failure is 2880 hours
- the time to shift a VM from one server to the other is 0,000005h
- the cloud is operational if at least one VM is operating.

The performability measure we are interested here is the probability of an unavailable software/hardware system state, which means that there is no VM running any more. This is expressed as (#runningVM1 + #runningVM2) == 0 in the syntax of the used tool TimeNET; # denotes the number of tokens in a place. The actual controlled train control system will become unsafe or non-operational if the cloud system is not available.

3 *MULTI-TRAJECTORY SIMULATION FOR RELIABILITY EVALUATION*

Several evaluation methods can be applied to the type of models introduced in Section 2.2. Numerical analysis creates the full reachability graph, and constructs and solves a Markov chain model after that. This is possible as long as the state space is not too large and there are no non-exponentially timed delays (or not too many in the case of extended non-Markovian models, which introduce additional algorithmic complexities [11]). Simulation is applicable without such restrictions, but may lead to unacceptably long run times: With the assumed MTTFs and MTTRs and level of fault-tolerance, our model setup is obviously a rare-event problem [5, 12] as the expected downtimes are very small. The numerical results covered in Section 4 for different parameter configurations compare methods accordingly. The multi-trajectory simulation method proposed in earlier work [15] allows to integrate elements of numerical analysis with simulation. One of the goals of this paper is to show that it can be configured to solve rare-event simulation problems like the one in Figure 2 efficiently.

The method works as follows: (for details refer to [15]) We introduce the notion of a particle that informally captures one entry in a sparse probability vector. A particle ϕ is thus a pair (m,w) with a marking (state) $m \in S$ and its corresponding weight, a probability $w \in [0, 1]$. The multi-trajectory algorithm follows this scheme: start with one simulation particle for the initial at simulation time $t = 0$ with weight 1. The algorithm maintains two sets of particles, namely the current Φ and next Φ'. In each step of the main simulation loop, the algorithm iterates over the current set of particles Φ. For each particle $\phi \in \Phi$, two treatments are possible: 1) the particle is simply followed as in a standard simulation by probabilistically choosing one of the subsequent states. The weight of the previous particle is kept (termed Move); or 2) *all* possible subsequent states are computed similarly to an iterative step in a numerical transient solution of a Markov chain (termed *Split*).

The weight of the particle is distributed over all descendant particles by multiplying it with the enabled transition's firing probabilities. It should be noted that the algorithm works in a discrete time scale, but transfers continuous-time models on the fly.

All created particles are stored in the next particle set Φ'. If a particle with an identical state (marking) already exists, their weights are simply added (the particles are merged, thus reducing the number of particles to be stored). The sum of all particle weights at any time step remains one. When the full particle set Φ has been iterated, a sample of the performance measure is generated based on the current particle vector, and Φ is set to Φ' to start the main loop again. Unbiasedness and convergence properties of this algorithm have been treated in [15].

A first general version of this algorithm, which is not directed towards rare-event simulation, has been implemented in TimeNET [17]. As pointed out in [15], the main issue towards using the method in rare-event settings efficiently is to control the particles such that the achieved speedup is maximized. Storing several particles increases the coverage of the reachability set, and maintaining weights instead of probabilistic decisions further enhances the chance to enter interesting areas in the state space, e.g., where the evaluated system is failed. Experiences from rare-event simulation methods suggest to differ between regions of states that have a similar probability to arrive at the state of interest in the simplest case [14, 16], and to split simulation paths such that there is a comprehensive coverage of the state space.

3.1 A New Region-Balancing Heuristic

How can this be transferred into a reasonable heuristic for our multi-trajectory algorithm? Imagine a set of regions *R* that partition the state space, which contain states with a similar importance for the performance measure (which can often be interpreted as how likely it is to reach a state of interest out of that region). We propose to transfer the idea of an importance function similar to the one used in splitting simulations [16].

The importance function of a state should estimate a function corresponding to the future impact of trajectories starting from that state on the performability result. Our multi-trajectory method can be used for any type of performance measure, but in the reliability setting of this paper, only one probability of being in a non-safe state has to be computed. Another simplifying advantage is that a fault-tolerant system such like the one shown in Figure 2 has a natural way of estimating the riskiness of a state of the model based on the number of failed sub components that did not yet lead to system breakdown.

The simple heuristics proposed in [15] only control the splitting of particles based on the available number of particles *N*. In this paper, however, we propose to emulate the idea of splitting simulations by trying to maintain similar population sizes of particles in all regions *R* by using at most *N* particles.

How can particle numbers be controlled in regions? The simple approach in [15] just decided for each particle whether to split or not. For an improvement with respect to reliability problems as in this paper, we propose a region-balancing extension by:

1) always split particles while there is at least 20% of particle space still available;

2) split particles irrespective of available space if they belong

to a region occupying less than 200% of its fair share (otherwise, empty regions would decrease the overall population based on our experiments) – this may increase the particle number temporarily above *N*;
3) after each full iteration of particles in Φ, check if the number of particles |Φ| exceeds the available amount *N* and reduce the size until it fits again.

To achieve such a reduction, we propose an unbiased probabilistic selection of particles populating one region: select k particles $p_1 \ldots p_k$ (out of an over-populated region r_i, i.e., in which the number of particles is > N / |*R*|) which should be diminished. Draw a probabilistic choice according to the relative probabilities (weights) of the particles, i.e., select one particle p_i with probability $w(p_i)/\sum_i w(p_.)$ with $w(p_i)$ denoting the probability of the particle. Remove the k particles from the particle set, and add a (selected surviving) particle with the state of p_i with probability $w(p_i) = \sum w(p_.)$.

4 ANALYSIS RESULTS FOR THE CASE STUDY

A prototype implementation of the region-balancing algorithm extension covered in Section 3 has been implemented in TimeNET [17] for this paper. Validation was done for models with known results such as a Petri net variant of an MM1K queueing system, and by comparing with results of numerical analysis of the tool.

The new algorithm variant is used here to evaluate the model depicted in Figure 2. Our goal is to show that the multi-trajectory algorithm, which is already considerably faster than regular simulations in many cases [15], can be sped up even more for rare-event problems by balancing the particle regions. Our experiences so far show that for simple-structured problems such as an MM1K model of a reliability evaluation, the blocking probability (i.e., P{queue full}) can be easily configured to a rare-event problem, and the algorithm is working as expected. It speeds up the simulation while resulting in very small relative errors when the importance function is chosen as the obvious #queue.

Reliability evaluation of the cloud server system of Section 2 with the algorithm turned out to be more challenging. Our first assumption is a simple heuristic importance function that takes into account the number of not currently running VMs, the number of failed PMs, and the number of failed CNs. The corresponding importance function is defined for the model as #failedPM1 + #failedPM2 + #failedCN1 + #failedCN2 + 2**A* - #runningVM1 - #runningVM2, as there could be at most 2**A* VMs running with the restriction of PMs. This function is evaluated for each Petri net marking (or particle), returning a value in the range 0..12 with higher values expected to be "closer" to the system failure. As there are only a few different values, we simply assume all particles with the same importance function value to belong to the same region of states. We expect a tradeoff between the number of regions and possible occupancy numbers in relation to the overall number of particles *N*, which should be investigated further in the future.

All calculations have been run on a Windows laptop with AMD Ryzen 7 CPU at 2.3GHz. Five simulation runs have been executed for each setting with different seed values, and the results are averaged over the different values. Simulations were stopped after 10 seconds run time and the remaining error is compared to assess the convergence speed (i.e., achieved accuracy vs the identical computation time budget). For the experiments covered here we chose parameter values $A = C = B = 2$, corresponding to two CNs, two PMs and four VMs per server. The regular simulation with just one particle without any splitting is not able to generate any usable non-zero result, which is typical for a rare-event problem. Moreover, it will easily exceed the memory requirements of numerical analysis techniques for higher values of *A*, *B* and *C*. The setting was chosen small here to have a comparison value, which the numerical analysis computes as 6.30E-6. The model has about 36 000 states, of which 17 993 are tangible (identical to the size of the system of linear equations to be solved).

The regular multi-trajectory algorithm set to "always split" [15] arrives at a result of 2.06E-6 within the short time budget, equaling a relative error of 67%. The new region-balancing version proposed in this paper results in an average unavailability of 6.36E-6, which corresponds to a relative error of only 2.5%. Our results show also for other settings that the extended algorithm is significantly more efficient (in terms of achieved accuracy per computation time) for the model problem. More in-depth evaluations of particle number tradeoffs are planned. For instance, it is not obvious what an optimal maximum particle number is, as larger numbers allow a better state space coverage, but take more time in the evaluations per step and the region size management.

5 CONCLUSIONS

The paper presented an extension of a multi-trajectory simulation algorithm aimed at reliability evaluation of complex dynamic systems. The advantage of the algorithm is its hybrid mix of analytical and simulation elements, which allows it to be configured to a wide range of application problems. The paper proposed a region-balancing method for reliability models that is based on ideas from rare-event splitting simulation methods. Moreover, a stochastic Petri net model for an industrial case study problem of a safety-critical train control system cloud server architecture is evaluated.

The applicability of the extended algorithm to systems with huge state spaces and the impact and selection of the importance function should be analyzed in future work.

ACKNOWLEDGEMENT

The authors acknowledge the work of Atieh Nazariasl for running the TimeNET evaluations of the model, and discussions on parameter configurations.

REFERENCES

1. K. Trivedi, A. Bobbio, "Reliability and Availability Engineering: Modeling, Analysis, and Applications", Cambridge University Press 2017.
2. B. Silva, E. Tavares, R. Matos, P. Maciel, A. Zimmermann, "Sensitivity Analysis of an Availability Model for Disaster Tolerant Cloud Computing System", *Int. J. Network*

Management, Issue 6, Vol. 28, 2018.
3. K. S. Trivedi, "Probability and Statistics with Reliability, Queuing and Computer Science Applications", 2nd ed. Hoboken: Wiley, 2002.
4. M. Malhotra, K. S. Trivedi, "Power-hierarchy of dependability-model types" *Reliability, IEEE Transactions on,* 1994 , vol. 43, pp. 493-502.
5. J. K. Muppala, R. M. Fricks, K. S. Trivedi, "Techniques for system dependability evaluation," in *Computational Probability, Int. Series in Operations Research & Management Science*, Springer US, 2000, vol. 24.
6. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, "Modelling with Generalized Stochastic Petri Nets", John Wiley and Sons, Hoboken 1995.
7. A. Zimmermann, "Stochastic Discrete Event Systems — Modeling, Evaluation, Applications", Springer, 2007.
8. M. Malhotra, K. Trivedi, "Dependability modeling using Petri-net based models," *IEEE Trans. on Reliability*, vol. 44 (3), 1995.
9. Analysis techniques for dependability — Petri net techniques, IEC 62551:2012, IEC Norm DIN EN 00 338, Sep. 2013.
10. Application of Markov techniques, IEC 61165:2006 Ed. 2.0, IEC Norm DIN EN 00 338, May 2006.
11. G. Ciardo, R. German, C. Lindemann, "A Characterization of the Stochastic Process Underlying a Stochastic Petri Net", *IEEE Transactions on Software Engineering* 20 (1994), pp. 506–515.
12. J. Faulin, A. A. Juan, S. Martorell, J.-E. Ramirez-Marquez, Eds., "Simulation methods for reliability and availability of complex systems", Springer 2010.
13. P. W. Glynn, D. L. Iglehart, "Importance sampling for stochastic simulations," *Management Science*, vol. 35, no. 11, (Nov.) 1989.
14. P. Glasserman, P. Heidelberger, P. Shahabuddin, T. Zajic, "Multilevel splitting for estimating rare event probabilities," *Operations Research*, vol. 47, pp. 585–600, 1999.
15. A. Zimmermann, T. Hotz, "Integrating Simulation and Numerical Analysis in the Evaluation of Generalized Stochastic Petri Nets", *ACM Trans. Modeling and Computer Simulation* (TOMACS), vol. 29/4, Nov. 2019.
16. A. Zimmermann, Paulo Maciel: "Importance Function Derivation for RESTART Simulations of Petri Nets", *9th Int. Workshop on Rare Event Simulation* (RESIM 2012), pp. 8-15, (Jun.) 2012.
17. A. Zimmermann, "Modelling and Performance Evaluation with TimeNET 4.4", *Proc. Quantitative Evaluation of Systems (QEST 2017) 14th Int. Conf., LNCS 10503*, (Sep.) 2017, pp. 300-303.
18. M. Jammal, A. Kanso, P. Heidari et.al.: „Evaluating High Availability-aware Deployments Using Stochastic Petri Net Model and Cloud Scoring Selection Tool", IEEE Trans. on Services Computing (Early Access), Dec. 2017.
19. J. Villén-Altamirano: "Importance Functions for RESTART Simulation of Highly-Dependable Systems", Simulation 83, 12 (2007), 821–828.

BIOGRAPHIES

Armin Zimmermann, PhD
Department of Computer Science and Automation
Systems and Software Engineering Group
Technische Universität Ilmenau
Helmholtzplatz 5
D-98693 Ilmenau, Germany

e-mail: armin.zimmermann@tu-ilmenau.de

Armin Zimmermann serves on the faculty of Computer Science and Automation at Technische Universität Ilmenau (Germany) since 2008, and as director of its Institute for Computer and Systems Engineering since 2012. Professor Zimmermann's research interests include stochastic discrete-event models for performance and reliability evaluation of embedded systems and their tool support. He is project leader of the TimeNET software tool since 1996 and author of a Springer book on Stochastic Discrete Event Systems. He earned his Diploma, PhD and Habilitation degrees in computer science at the Technische Universität Berlin, and was a visiting researcher at the University of Zaragoza, the University of Potsdam and the University of Canterbury at Christchurch.

Thomas Hotz, PhD
Department of Mathematics and Natural Sciences
Group for Probability Theory and Mathematical Statistics
Technische Universität Ilmenau
Weimarer Str. 25
D-98693 Ilmenau, Germany
e-mail: thomas.hotz@tu-ilmenau.de

Professor Thomas Hotz studied Mathematics at the University of Heidelberg until 2002. He then worked as a medical statistician for one year at the University of Leicester, UK. From there he moved to New York to work at the United Nations Statistics Division for another year before returning to academia. In 2007, he received a doctoral degree from the University of Göttingen. In 2012, he joined the TU Ilmenau where he is Head of the Group for Probability Theory and Mathematical Statistics. His research interests include a wide variety of topics in applied and mathematical statistics, in particular concerning the analysis of complex data objects.

Volker Hädicke
Thales Deutschland GmbH
Thalesplatz 1
71254 Ditzingen, Germany

e-mail: volker.haedicke@thalesgroup.com

Volker Hädicke acts as the safety manager in the referenced project. He has been working as RAM and safety manager for over 20 years in various areas of technical safety, especially in the field of railway safety.

Martin Friebe, PhD
Thales Deutschland GmbH
Thalesplatz 1
71254 Ditzingen, Germany

e-mail: martin.friebe@thalesgroup.com

Martin Friebe works as a RAM and Safety Manager at Thales in the Ground Transportation division. Previously, he has held similar positions in the naval and land defense sector. He earned his doctoral degree in Naval Architecture and Ocean Engineering at the University of Tasmania in 2020. He earned his bachelor in Mechanical Engineering and his master's degrees in Ocean Engineering at Seoul National University in South Korea. Mr Friebe's research activities are centered around survivability failure mechanisms for naval combat vessels, whereas his most recent activities are focused on RAMS for railway signaling systems.