IEEE 11th Int. Systems Conference (SysCon 2017), Montreal, April 2017

Model-Based Dynamic Reliability Engineering for Hybrid Electric Vehicle Design

Armin Zimmermann and Thomas Dietrich
Systems and Software Engineering
Computer Science and Automation Dept.
Technische Universität Ilmenau
Ilmenau, Germany
armin.zimmermann@tu-ilmenau.de

Paulo Maciel
Modeling of Distributed
and Concurrent Systems
Federal University of Pernambuco
Recife, Brazil
prmm@cin.ufpe.br

Andreas Hildebrandt
Professional Training and
Functional Safety Engineering
Pepperl+Fuchs GmbH
Mannheim, Germany
ahildebrandt@de.pepperl-fuchs.com

***Abstract*—The paper analyzes the tradeoff between battery life and functional reliability for a hybrid electrical vehicle design study. Reliability of complex technical systems often depends on their dynamic behavior. Petri nets have significant advantages over classic static models of reliable systems in describing such systems, and an international standard IEC 62551 has been published recently (Analysis techniques for dependability — Petri net techniques). Simulation is the only possible method to derive reliability results for realistic systems that have a large state space or are not Markovian (memoryless). The paper shows how the very long run times for highly reliable systems can be sped up significantly by using the RESTART-method for stochastic Petri nets and its implementation in the TimeNET software tool.**

I. Introduction

The systematic design of complex technical systems requires prediction of non-functional properties of a planned architecture in many fields of engineering applications. This is especially hard but important in early design phases, as such properties often depend on the overall system composition (they are "emergent") and can thus be measured only after the complete system has been built (at least as a prototype). 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 such as in train control or avionics.

An area with increasing importance of engineering systems design, both in terms of safety and cost, are hybrid electric vehicles (HEV). Such cars can help to lower the use of fossil fuel and decrease air pollution. The resource-efficient operation of HEV has several constraints and requirements including energy use, battery life time and reliability, and is only possible with a complex control software and high effort for design, parameter setting and testing. Model-based systems design with the evaluation of performance and reliability measures are therefore important for early design phases to avoid costly redesigns.

Design decisions should be based on the analysis of the effect of variants and parameters, as long as no prototype is available. There are many models available in the literature for reliability evaluation [1], [2]; their industrial use is often restricted to static models including fault trees and reliability block diagrams. The reliability of complex technical systems is, however, often achieved by fault-tolerance and reconfiguration, or in general depends on the dynamic behavior [3].

The joint behavior of a controlled system, determined by its controller and changing environment, is often significant for reliability results (as demonstrated by the xase study in this paper). Without taking these system elements into account, resulting models and results will be unrealistic or unnecessary conservative, and thus the best system design is not found.

In the literature, model-based systems design for electric vehicles is more often done based on continuous or hybrid models, as they are well-established in electrical engineering and control theory. For example, a controller design of a series HEV design is proposed in [4]. Aspects of energy management for HEV have been covered in [5], and its combination with drivability issues in [6]. Model-based systems optimization for HEV is conisdered in [7], [8], [9]. The reliability of HEV setups has been analyzed with MATLAB models in [10], and the tradeoff between reliability and fuel economy in [11]. A hybrid fluid Petri net is used in [12] to analyze the effect of electric vehicle charging on network stability. However, efficient rare-event simulation techniques for reliability properties of continuous or hybrid models are not available so far.

Petri nets [13] and their stochastic timed extensions are well-established for dynamic, distributed systems and their reliability evaluation [14], [15]. Their industrial use is supported by the recent norm IEC 62551[16] (Analysis techniques for dependability — Petri net techniques). Simpler dynamic models with memory-less stochastic behavior can be analyzed with Markov chains [17], [1].

Models with a large state space or non-Markovian delay distributions require simulation methods for their evaluation [18]. 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 the most prominent techniques to tackle it are *importance sampling* [19] and *splitting* [20]. Both strategies enforce certain trajectories in the simulation, to get

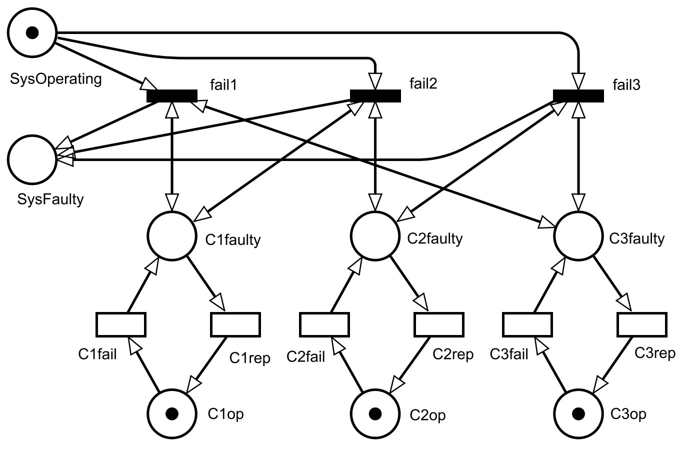

Fig. 1. Petri net model of a 2-out-of-3-system model according to IEC 62551

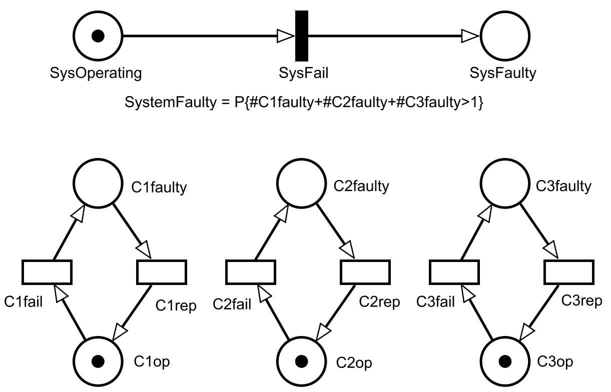

Fig. 2. Proposed simplified 2-out-of-3-model

more samples of interest with the same amount of simulated events. The results have to be transformed to take the changes into account.

Splitting methods have the advantage of being less dependent on experts able to change the model, and not requiring a purely Markovian model. Proposed algorithms such as RESTART [21] are easier to implement independently of the model, and there are recent results on the derivation of its parameters for Petri nets [22].

This paper describes the advantages of Petri nets for the reliability modeling and evaluation of complex dynamic systems in the automotive area. The subsequent section briefly introduces stochastic Petri nets along a reliability model from the standard IEC 62551. A simplified reliability modeling pattern is proposed. Section III gives a short overview of the RESTART method to speed up reliability simulations. Our software tool TimeNET is covered afterwards, which implements analysis and rare-event simulation techniques for stochastic Petri nets. Section V introduces an HEV case study. Its systems design tradeoff between reliability and battery life can be analyzed by a Petri net as shown in this section. The presented Petri net model is a significant simplification compared to the first introduction of this example [23]). The efficiency of the speed-up simulation technique is presented in the results section.

The contribution of the paper include reduced modeling patterns w.r.t. the relevant standard, a simplified HEV Petri net model, and the application of rare-event simulation techniques to automotive systems design.

II. Reliability Modeling with Stochastic Petri Nets

This section briefly covers stochastic Petri nets; for more details the reader is referred to the literature (e.g. [14], [16], [24]). The explanations follow the small example model shown in Figure 1, which represents the IEC 62551's proposal how to model a fault-tolerant, redundant 2-out-of-3 component system.

A *stochastic Petri net* can be specified as a tuple $\mathsf{SPN} = (P, T, \vec{Pre}, \vec{Post}, \Lambda, W, \vec{m}_0)$ [24]. *Places* $p_i \in P$ are depicted as circles and correspond to the discrete-valued state variables of the system, for which the current value is shown as a number of *tokens* (black dots or number) inside. In our example the places `SysOperating` and `SysFaulty` represent the status of the overall system, while the lower places `C1op` and `C1faulty` (as well as their counterparts for the other components C2 and C3) model the local states of the redundant components. The *marking* $\vec{m}$ of a Petri net assigns a natural number of tokens to each place $\vec{m} : P \to \mathbb{N}$, and is thus equivalent to the state of the model (and thus the state of the underlying stochastic process during its dynamic evaluation).

The set of all theoretically possible *markings* (not all of them are actually reachable in many cases) is denoted by $M = \{\vec{m} \mid \forall p \in P : \vec{m}(p) \in \mathbb{N}\}$. The graphical picture of a Petri net usually shows the *initial state* $\vec{m}_0 \in M$ of the system. For our example in Figure 1, all components are operable at the start of an analysis, and thus $\vec{m}(\texttt{C1op}) = 1$, for instance.

The set T contains the *transitions*, i.e., the active elements of the model, which are depicted as rectangles of different kinds depending on their type. `C1fail` and `C1rep` model the failure and repair actions of the first component. A model element cannot be both a place and a transition ($T \cap P = \emptyset$), as a Petri net is a bipartite graph, and a meaningful model should not be empty $T \cup P \neq \emptyset$ [13].

Arcs, that change the state when a transition fires, are directed connections between places and transitions (and vice versa). They describe the prerequisites for the activation (enabling) of a transition as well as the change in the state variables as a consequence of the activity that is modeled by the transition. *Input arcs* $\vec{Pre} : P \times T \to \mathbb{N}$ connect places with transitions, and specify the multiplicity (the default of 1 is not shown in figures). If there is no arc from p_i to Transition t_j, then $\vec{Pre}(p_i, t_j) = 0$. Multiplicities of *output arcs* from transitions to places are given by $\vec{Post} : P \times T \to \mathbb{N}$. Sometimes figures contain arcs looking like having arrow heads in both directions; this just means there are two arcs (one input and one output) between the same transition-place pair drawn on top of each other, which emulates a test arc that does not change the number of tokens in a place.

The *firing delay* Λ of a transition specifies the time that needs to pass after the modeled activity starts until it is finished and the transition fires. This time can be stochastically distributed according to a certain distribution function. Special cases are *immediate transitions*, which fire without delay ($\Lambda(.) = 0$) and are depicted as filled, thin rectangles. (Example: `fail1` in Figure 1, which fires when components 1 and 3 have failed). W specifies for all immediate transitions the *firing weight*, a real-valued number that corresponds to the relative firing probability of the transition if there are several of this type enabled at the same time.

The semantics (or dynamic behavior) of a stochastic Petri net is defined by the enabling and firing rule. A transition t is said to be *enabled* in a marking $\vec{m}$ if there are enough tokens in its input places $\forall p \in P : \vec{m}(p) \geq \vec{Pre}(p, t)$.

When a transition becomes enabled, a sample random value is drawn from its firing time distribution, and assigned to it as a *remaining firing time*. These times decrease for all transitions at the same speed[1]. The stochastic process remains in a state until the first remaining firing time reaches zero.

The fastest transition t (if there are several at the same time, a probabilistic choice is done) *fires*, and changes the current state of the Petri net $\vec{m}$ to a new one $\vec{m}'$ by decreasing the number of tokens in the input places and adding tokens to the output places $\vec{m} \xrightarrow{t} \vec{m}'$ with $\forall p \in P : \vec{m}'(p) = \vec{m}(p) + \vec{Post}(p, t) - \vec{Pre}(p, t)$.

There are many extensions to this basic definition in the literature [24]. Input arcs specify dependencies of an action that are consumed when this is finished. Sometimes this behavior is not wanted, and the restricting conditions of an action should not change. *Inhibitor arcs* and *guards* can be used then. The former are connections from a place to a transition with a multiplicity $Inh : P \times T \rightarrow \mathbb{N}$ ending with a small circle (c.f. Figure 6), and disable the transition when there is one (or the specified number of) tokens in the place: An additional requirement for transition t to be enabled in marking $\vec{m}$ is then: $\forall p \in P : \big(Inh(p, t) > 0\big) \longrightarrow \big(Inh(p, t) > \vec{m}(p)\big)$.

Guards represent a similar, but more expressive concept: Only if the Boolean guard function $G : T \times M \mapsto \{\text{True}, \text{False}\}$ evaluates to True for a given marking $\vec{m}$, the transition is enabled: $G(t, \vec{m}) = \text{True}$.

The standard on reliability modeling with stochastic Petri nets [16] proposes model patterns with immediate transitions to emulate Boolean reliability dependencies which are conventionally used in fault trees and reliability block diagrams. This leads to dynamic modeling of processes that are not present in the real system, but stem from workarounds that derive the wanted place markings just like a computer algorithm. We propose to an expression of such aspects with measure expressions, guards, or inhibitor arcs, because they are closer to the modeled issues. The model shown in Figure 1 can then be simplified to Figure 2, where the upper part of immediate transitions is removed by simply specifying the reliability measure of interest as it has been done with `SystemFaulty`[2]. Models will be much easier to understand when they are restricted to actual system behavior. Moreover, as an example, more general m-out-of-n redundant systems are close to impossible to specify with the method of [16], as all combinatorial possibilities would have to be modeled. If there is still a good reason to explicitly model the overall system state (operational / failed) with places, this can be done in a reduced way as shown in Figure 2 with a transition `SysFail` and a guard function `#C1faulty + #C2faulty + #C3faulty > 1`.

III. Computationally Efficient Simulation of Reliability Models

Performance and reliability evaluation of complex dynamic systems on the basis of stochastic Petri net (or other) models is usually only tractable with simulation methods [16]. Unfortunately, statistically sound predictions for highly reliably systems require very long simulation times to generate enough events of interest.

Speed-up methods for such simulations can shorten this time by several orders of magnitude. Splitting methods and especially RESTART (*repetitive simulation trials after reaching thresholds*, [21]) have the advantages of being relatively independent of the underlying model and being robust against influences of the algorithm parameters. They require the specification or derivation of a heuristic *importance function* $f_I(\vec{m})$, which returns greater values for states $\vec{m}$ that are closer to the rare region of inteerst, i.e., the probability of a simulation trajectory to reach a state that contributes significant sample(s) is higher from there. This assumes that there are paths towards such states and that overall system failures, for instance, do not happen without prior deterioration. This assumption is true for many complex, redundant systems in which partial system failures, aging, or wear and tear increase the probability of a system failure.

Figure 3 sketches the method of splitting: assume that state in the region $A = RS_3$ are of interest (rare failure of a highly reliable system, for instance), and the low probability of being in or reaching this state set shall be computed. The set of all reachable states is denoted by RS_0, which is partitioned into subsets $RS_1 \ldots RS_L$ such that $A = RS_L$ and $RS_L \subset RS_{L-1} \subset \ldots \subset RS_1 \subset RS_0$.

A state's membership in one of the subsets RS_i is specified by the importance function $f_I : RS_0 \rightarrow \mathbb{R}$ together with a set of threshold values $Thr_i \in \mathbb{R}, i = 1 \ldots L$. The subsets are then defined by $RS_i = \{\vec{m} \in RS_0 \mid f_I(\vec{m}) \geq Thr_i\}$. A state $\vec{m}$ belongs to a threshold i if and only if $\vec{m} \in RS_i \setminus RS_{i+1}$.

The general idea of splitting is the following: instead of estimating the overall probability $\text{P}\{A\} = \text{P}\{RS_L\}$ of being

[1]Assumption of a global clock, which is fine for engineering applications but obviously not in line with Einstein's special theory of relativity if there is relative motion between the system elements.

[2]If the components have identical times for failure and repair, the sub models can even be merged into an integrated one with three tokens and transitions with firing semantic *infinite server*.

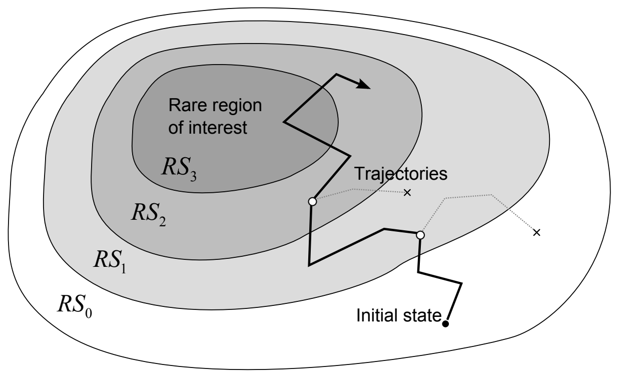

Fig. 3. Simulation trajectories in the state space under splitting

in a state in RS_L (which is computationally hard), estimate the conditional probabilities of reaching the next thresholds from each region $\mathrm{P}\{RS_{i+1} \mid RS_i\}$. The final result is then computed as the product of the set of conditional probabilities, which are much easier to estimate via simulation.

A standard simulation follows trajectories in the state space that are likely with a high probability, as it emulates the real behavior. With the RESTART algorithm, the importance function is monitored for each trajectory, which is split into several trajectories if a new region is reached (i.e., the importance function of a newly reached state exceeds a threshold). Trajectories for which a threshold falls below a threshold are canceled with the exception of the last remaining one of each split, to discard unsuccessful trials. The algorithm simply stores the current state for each newly reached region (small circles in Figure 3), to restart in a backtracking-like fashion from there again when the current trajectory fell under the threshold. The figure sketches this behavior for splitting each trajectory into two; these *splitting factors* R_i should be set in general such that one trajectory on average survives the step up to the next threshold [25]. The simulation works like normal in the regions RS_0 and $RS_L = A$.

Such a changed simulation needs correction factors that incorporate the trajectory splitting. In our implementation, each trajectory is assigned a current weight ω which is initially set to 1 and is divided by R_i upon each split into R_i trajectories. When a trajectory is the last surviving one of its generating split and falls below threshold i, its weight is multiplied by R_i. Figure 4 sketches this behavior, which ensures that the reciprocal of each trajectory's weight equals the average number of simulations that would have been required to start to get one trajectory to this region.

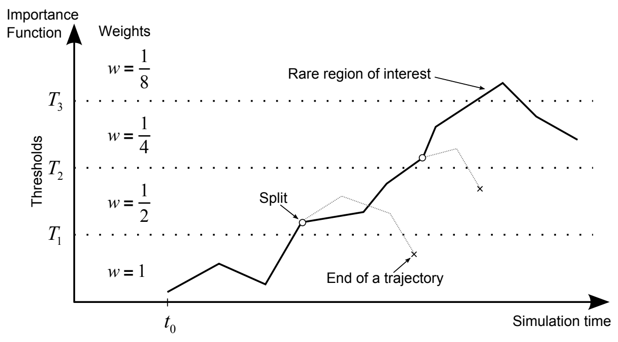

Fig. 4. RESTART importance function, thresholds and weights

A simulation collects partial results contributing to the computation of a performance/reliability measure during its execution, for which a weighted average represents the current estimate of the measure. Such *reward measures* may depend on states and events (state changes), and specify a function F defined on the stochastic process (as defined by the model and its semantics) including its current state t. To incorporate the splitting, the only necessary change is to multiply partial contributions (rewards) by the current weight $\omega(t)$ of a trajectory when results are collected. A stationary simulation then computes an estimation of the reward measure F with

$$\lim_{T\to\infty} \frac{1}{T} \int_0^T \omega(t)\, F(t)\ \mathrm{d}t,$$

with T denoting the simulation time.

There are several possible variants of such splitting algorithms characterized in the literature [26]. There are theoretical results for the optimal speedup with certain settings of thresholds and importance function [21], which will rarely be possible in reality, but experiences show that the method is quite robust against differences and that the achievable speedup is usually several orders of magnitude. Recent results show that the necessary parameters of a RESTART algorithm can be derived from the Petri net structure in an automatic way [27], [28], which is a principle advantage over less structured models such as Markov chains or simulation programs.

IV. Software Tool TimeNET

Modeling and evaluation of stochastic Petri nets is supported by several software tools. The paper uses TimeNET [29], which offers uncolored *extended deterministic and stochastic Petri nets* (eDSPN [30]) and colored stochastic Petri nets (CSPN [24]) as well as other model classes aiming more to research and educational purposes. Several stationary and transient analysis and simulation algorithms are availabe in the tool. TimeNET has been used in the IEC 62551 standard document for the analysis of example case studies [16]. The RESTART method for rare-event simulation speedup has been implemented both for eDSPNs and CSPNs in the tool. To the best of the authors' knowledge, it is the only available tool supporting rare-event simulation of colored Petri nets.

TimeNET is currently available for Windows and Linux (in 32 and 64 Bit versions for both), and can be downloaded free of charge for non-commercial use from `http://www.tu-ilmenau.de/TimeNET`. An overview of other stochastic Petri net tools can be found in [31] as well as via the tools list of the Petri net home page[3].

[3]http://www.informatik.uni-hamburg.de/TGI/PetriNets

V. Modeling and Evaluation of an HEV

An application study for the model-based analysis of a hybrid electric vehicle is presented in this section. Such vehicles can be classified depending on their degree of integration of electrical power functions from micro over mild to full hybrid. We consider a *full hybrid* car design here, being characterized by the possibility to fully run on the electric motor for some time span (as opposed to using it only as an addition). The internal combustion engine (ICE) as well as the electric motor can be run together or either one alone. The way how both propulsion methods are mechanically integrated allows varying powertrain architectures: *parallel, series*, and *power-split* designs are among the most important variants [32].

We consider a parallel full hybrid design in the following, being the most common design type currently[4]. Its general architecture is sketched in Fig. 5; the underlying system assumptions as well as the later driver behavior have been adapted from [23].

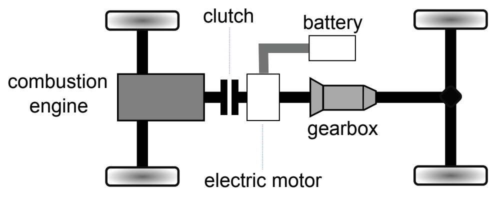

Fig. 5. Simplified structure of a parallel full hybrid following [23]

This type of HEV design can operate in several modes: as long as battery capacity and power/torque available from the electric motor are sufficient, a pure electric drive (*e-drive*) mode can be used with the ICE switched off and thus no exhaust. The battery has to be charged when it's stored energy (*state of charge*, SOC) drops below a certain threshold. The ICE is started and runs at a higher torque than needed to cruise or accelerate the car at that moment. The electric motor is used as a generator to create electric energy to be stored in the battery (*charge mode*). ICE and electric motor can also be operated together to increase the overall available power, usually during short accelerations (*boost mode*). Kinetic energy is (in part) transformed back into electric energy during braking, when the electric motor is applied as a generator (*regenerative braking*). It should be noted that this is a simplified model of the base setup; the best-selling HEV, the Toyota Prius, has 6 modes with different mixes of the power sources.

The battery is often the most expensive part of an HEV, and its life span depends heavily on the number of charge/discharge cycles, the levels of discharge, as well as their chemical type. Lithium-ion and nickel-metal hydride (NiMH) batteries are most widely used today, the latter for instance in the second-generation Prius. A computer controller decides about the operation mode depending on the current requested power and battery SOC. High and low values of SOC significantly decrease battery life; controllers thus impose shallow cycling to avoid them. In fact, current NiMH HEV batteries should be operated within an SOC range of $40\% \ldots 80\%$. This leads to only a portion of storable electric energy being available for HEV operation, a classic trade-off in systems design in which combustion efficiency (with higher stored amounts of electrical energy) has to be balanced against battery life time [8], [11] .

[4]A supervisory control system for a series hybrid is, for instance, considered in [4]

The non-trivial dependencies between operation mode, charge control, system reliability, and battery life span cannot be judged well without a model-based analysis. Driver behavior and environmental influences add stochastic elements. An according HEV system model will be presented in the next subsection. It is the basis for the later analysis of performance and dependability properties and their dependency on parameter settings in the controller. Numerical assumptions and driving behavior for our example have been taken from [23]. Differing from the analysis of several battery sizes we adopt a 1310 Wh battery here, as it is commonly used in the Toyota Prius.

A. A Stochastic Petri Net Model of an HEV

The application study is modeled with an (uncolored) stochastic Petri net in this section, to evaluate reliability measures afterwards. A similar model has been developed in [23] with a colored Petri net in the software tool REALIST [33]. Our study shows how the relevant aspects can be captured in a simpler and actually smaller model, allowing a more efficient performance and reliability analysis.

The model is shown in Figure 6 with the modules of the behavior grouped to increase understandability. The driver behavior / driving phases model is depicted left, as it influences the other parts directly or indirectly. Our considered operation modes of the HEV can be found in the center, while the rightmost part specifies the battery model with energy consumption and the state-dependent controller.

The driver behavior model is similar to an automaton: states correspond to marked places of the Petri net. The vehicle is parking at the beginning of a considered drive cycle (a token is in place `Parking`). The further places `Maneuvering`, `IntraUrban` und `ExtraUrban` model the later phases of backing into or reversing out of a parking space, driving inside a city, and on a highway. There are six transitions modeling the corresponding state changes, each with a firing delay according to the assumed length of driving phase (values taken from [23]). The places `inbound` and `outbound` ensure that the phases follow in the expected sequence: the vehicle token moves towards the lower places first, before going upwards again to the parking state.

The current driving phase influences the operation mode model in the middle. The three states cruising, braking and accelerating are captured with the places `Cruising`, `Decel` and `Accel`. This model module is empty during parking, the start of the vehicle creates a token in `Cruising`. The four transitions between the places model the state transitions, with the firing delays again chosen as in [23].

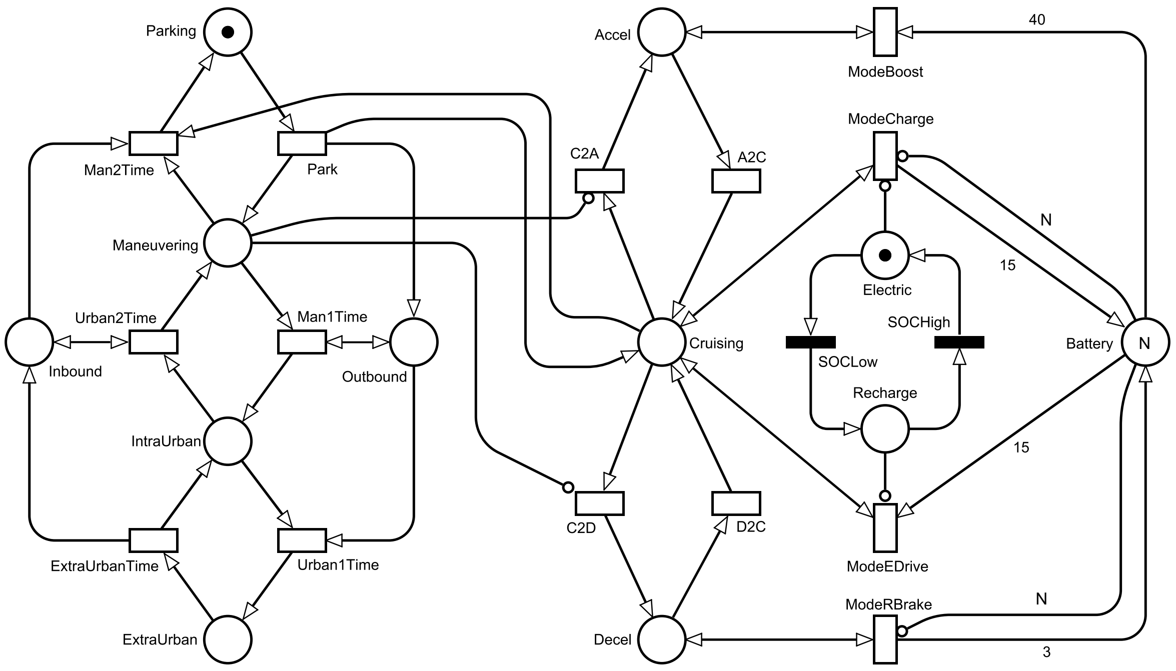

Fig. 6. Stochastic Petri net model of hybrid electric vehicle and driver behavior

Driving in the city and on a highway differs by the according frequencies of accelerations and braking. This is achieved by a marking-dependent firing delay of transitions `C2A` und `C2D`. The assumption of [23], that during maneuvering the vehicle is operated only electrically, is achieved with inhibitor arcs here.

The right part of Figure 6 contains the important place `Battery`, which models the current SOC with its amount of contained tokens. As we use a state-discrete model here (as opposed to control theory models mentioned in the introduction), this physical quantity has to be transformed with a certain discretization. There is a tradeoff between high accuracy (with high numbers of tokens per watt-hour) and efficient reliability evaluation, and in most software tools there is a maximum number of tokens that should be used. In our experiments (presented in the following subsection) a good setting turned out to be 100 watt-seconds equaling one Petri net token, which is assumed in the following. Model time is interpreted as in seconds, as the model itself is free of physical units of measurement. The model parameter N is then set as 37 728 tokens for a 1310 Wh battery at the maximum SOC of 80%.

Energy consumption and generation by the HEV during the different operation modes are modeled with the four transitions named `Mode...`. Braking (place `Decel` contains a token) leads to the repeated firing of transition `ModeBrake`, which adds three tokens upon each firing if the battery is not yet at its maximum SOC of 80%. The firing delay is chosen as 0.2 (seconds), setting the average brake-induced energy generation to 1.5 kW [23]. The other transitions have a firing delay of 0.1, according to 40 kW and 15 kW power generated or consumed.

An acceleration or braking by the driver uniquely defines the operation mode. During cruising, however, the controller sets the operation mode depending on the SOC. This is an extension over [23], which is geared to the controller according to the standard [16]. The other model parts correspond to the uncontrollable plant model. The controller is either in the state `Electric`, if the SOC is high enough, or in state `Charge`, in which the ICE is running and loading the battery. The setpoints of this very simple controller can be chosen with the transitions `SOCLow` and `SOCHigh` in our model, for instance firing when there are less than 18 864 or at least 37 728 tokens (for the SOC interval of $40\% \dots 80\%$). This behavior is achieved with *transition guards* (i.e., Boolean functions on the marking), but could also have been implemented by inhibitor and normal arcs. An instant reaction is achieved by using immediate transitions.

Exponential transition delay distribution have been arbitrarily chosen in the model, but non-Markovian delays would pose no problem as the evaluation is later done with simulation. The presented model is compact in comparison to the one introduced in [23]: it contains only 28 places and transitions, while the earlier model needed 69 despite being a more expressive colored Petri net.

B. System Analysis and Reliability Evaluation

The presented model can now be used to analyze the influence of the system's design parameters and controller settings on performance and reliability measures of the HEV. The results can be used to choose a balanced tradeoff between optimal battery control (SOC between 40% and 80%) and safety / reliability as experienced by the driver already during design. An example safety issue is the unexpected or unavailable reaction of the vehicle to driver requests, for

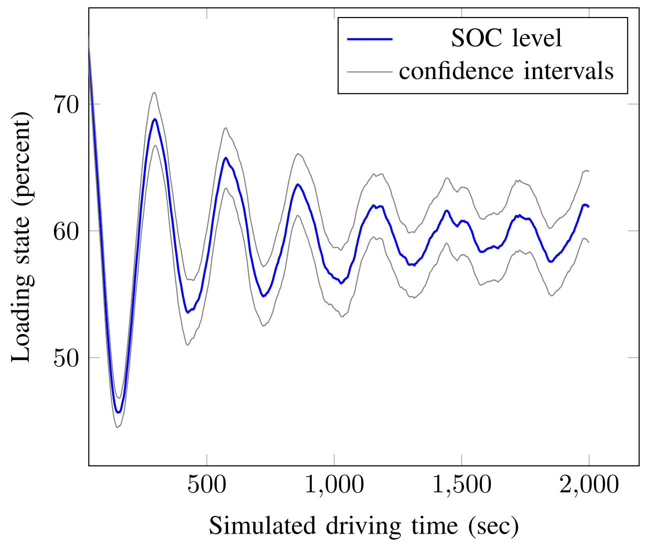

Fig. 7. Transient battery SOC over simulation time

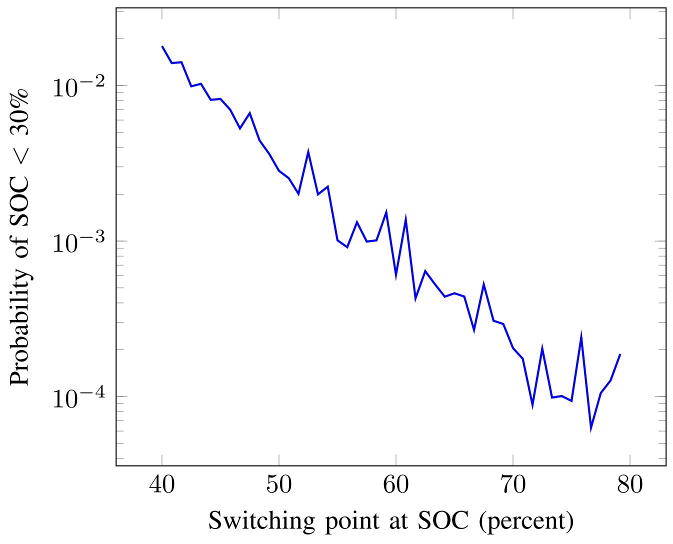

Fig. 8. Influence of switching point on low-battery probability

instance when there is not enough supplemental battery energy available during an acceleration for taking over another car, and thus the boost mode cannot be started or has to be canceled in between. Battery operation has been prioritized in the analysis example of [23]. In reality, the safety implications will probably have a higher importance, as small deviations from the optimal battery control and the resulting decrease in life span may be acceptable compared to safety-relevant driving functions. In this paper we thus analyze the opposite setup: driver commands are executed as far as possible, and the resulting effects on battery SOC is evaluated.

All computations have been carried out with the software tool TimeNET mentioned in Section IV on a laptop computer with Intel Core i7-4600U CPU running at 2.1 GHz. The stop criterion for the simulations has been set to an estimated confidence level of 95% and a relative error of 5%. To do this, the simulation computes mean and variance of the samples during the computations. Moreover, it derives the confidence interval size for an assumed normal distribution with the currently achieved variance (which is decreasing for a longer simulation run). The difference between one of the interval boundaries to the mean is finally compared to check if the relative difference is already small enough to stop.

The first experiment evaluates the controller's effect on operation and battery during a transient simulation over a time interval of $[0, 2000]$. The resulting SOC levels (averaged over many simulation runs until the specified accuracy has been achieved) are depicted in Figure 7. The specification of the corresponding performance measure in the tool can be simply done as `E{#Battery}` for the expected value of the number of tokens in the battery SOC place and the subsequent transformation back into percent.

The plot shown in Figure 7 visualizes the behavior of a naïve controller implementation, which switches only at the boundary SOC levels that it should ensure. The probability of actually going lower than 40% is thus quite high. The corresponding probability measure is specified by `P{#Battery<18864}` for the number of tokens equivalent to 40%. In reality, the controller should switch back into a mode that recharges the battery (or at least does not lower its SOC any further) earlier, i.e., at a higher SOC level for some safety margin. The functional dependency between the probability of a SOC lower than 40% and the controller switching point is analyzed with the experiment feature in TimeNET, that executes simulations for varying switching parameter settings in the range $40\% \ldots 80\%$. The resulting probabilities for low SOC values are shown in Figure 8; an almost logarithmic dependency is visible.

Critical measures for simulations, i.e., in which the run time becomes unacceptably long as covered in SectionIII, are very small probabilities which are typical for highly (safety-)critical systems or states. A possible variant of our case study example would be a mild hybrid HEV setup, in which the battery will always be recharged unless except for the acceleration boost mode. Very low battery SOC values should be very unlikely then. A variant of the presented model has been derived with parameters for the assumption that very low battery states such as SOC $< 30\%$ are rather critical for battery life, but that until a SOC level as low as 20% accelerations would still be supported electrically because of safety considerations. The probability of being in a state $20\% \leq \text{SOC} \leq 30\%$ (equal to the relative amount of time spent in such a state) has been evaluated and results in 1.38749E-5. The standard discrete-event, time-warp simulation of TimeNET took 3183 seconds CPU time, while the rare-event speedup algorithm RESTART needed only 272 seconds. Experiences with other applications (c.f. [34], [35], [22]) show that considerably higher speedups are typically achieved for "harder" problems, i.e., in which the result value is smaller by several orders of magnitude.

VI. Conclusion

Stochastic Petri nets are a valuable tool for model-based performance and reliability evaluation in a model-based systems design process. The corresponding standard IEC 62551 suggests possible steps and patterns for reliability modeling. However, efficient methods for the numerical computation of reliability measures for complex safety-critical systems are still an open field of research. Real-life industrial applications often violate the restrictions of direct methods, leaving only simulation for the derivation of result measures. Standard simulation is unfortunately too slow to estimate rare-event probabilities of highly reliable systems.

The paper presented the application of the RESTART algorithm to speed up such analyses. A model for a hybrid electric vehicle is presented that captures the necessary information much more compactly than an earlier proposal, and quantifies the effect of system design parameter settings on reliability measures w.r.t. battery life span. Future work will extend our methods further towards the automatic derivation of RESTART algorithm parameters and extending the model classes that the approach can be applied to.

References

[1] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, 2nd ed. Wiley, 2002.

[2] B. Bertsche, *Reliability in Automotive and Mechanical Engineering*. Springer, 2008.

[3] K. Kolowrocki and J. Soszynska-Budny, *Reliability and Safety of Complex Technical Systems and Processes: Modeling - Identification - Prediction - Optimization*, ser. Springer Series in Reliability Engineering. Springer, 2013.

[4] W. Shabbir and S. A. Evangelou, "Exclusive operation strategy for the supervisory control of series hybrid electric vehicles," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 6, pp. 2190–2198, Nov 2016.

[5] X. Zeng and J. Wang, "A parallel hybrid electric vehicle energy management strategy using stochastic model predictive control with road grade preview," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 6, pp. 2416–2423, Nov 2015.

[6] T. Miro-Padovani, G. Colin, A. Ketfi-Chrif, and Y. Chamaillard, "Implementation of an energy management strategy for hybrid electric vehicles including drivability constraints," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 8, pp. 5918–5929, Aug 2016.

[7] X. Hu, S. J. Moura, N. Murgovski, B. Egardt, and D. Cao, "Integrated optimization of battery sizing, charging, and power management in plug-in hybrid electric vehicles," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 3, pp. 1036–1043, May 2016.

[8] M. L. Shaltout, A. A. Malikopoulos, S. Pannala, and D. Chen, "A consumer-oriented control framework for performance analysis in hybrid electric vehicles," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 4, pp. 1451–1464, July 2015.

[9] A. A. Malikopoulos, "A multiobjective optimization framework for online stochastic optimal control in hybrid electric vehicles," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 2, pp. 440–450, March 2016.

[10] Y. Song and B. Wang, "Quantitative evaluation for reliability of hybrid electric vehicle powertrain," in *Power Engineering, Energy and Electrical Drives (POWERENG), 2013 Fourth International Conference on*, May 2013, pp. 1404–1409.

[11] M. A. Masrur, "Penalty for fuel economy — system level perspectives on the reliability of hybrid electric vehicles during normal and graceful degradation operation," *IEEE Systems Journal*, vol. 2, no. 4, pp. 476–483, Dec 2008.

[12] J. Hüls and A. Remke, "Coordinated charging strategies for plug-in electric vehicles to ensure a robust charging process," in *Proc. 10th Int. Conf. on Performance Evaluation Methodologies and Tools (VALUETOOLS 2016)*, Taormina, Italy, oct 2016.

[13] W. Reisig, *Petri nets*. Springer Verlag Berlin, 1985.

[14] M. Malhotra and K. Trivedi, "Dependability modeling using Petri-net based models," *IEEE Trans. on Reliability*, vol. 44, no. 3, pp. 428–440, 1995.

[15] J. K. Muppala, R. M. Fricks, and K. S. Trivedi, "Techniques for system dependability evaluation," in *Computational Probability*, ser. International Series in Operations Research & Management Science, W. K. Grassmann, Ed. Springer US, 2000, vol. 24, pp. 445–479.

[16] *Analysis techniques for dependability — Petri net techniques*, IEC 62551:2012, IEC Norm DIN EN 00 338, Sep. 2013.

[17] *Application of Markov techniques*, IEC 61165:2006 Ed. 2.0, IEC Norm DIN EN 00 338, May 2006.

[18] J. Faulin, A. A. Juan, S. Martorell, and J.-E. Ramírez-Márquez, Eds., *Simulation methods for reliability and availability of complex systems*. Springer, 2010.

[19] P. W. Glynn and D. L. Iglehart, "Importance sampling for stochastic simulations," *Management Science*, vol. 35, no. 11, pp. 1367–1392, Nov. 1989.

[20] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic, "Multilevel splitting for estimating rare event probabilities," *Operations Research*, vol. 47, pp. 585–600, 1999.

[21] M. Villén-Altamirano and J. Villén-Altamirano, "Analysis of RESTART simulation: Theoretical basis and sensitivity study," *European Transactions on Telecommunications*, vol. 13, no. 4, pp. 373–385, 2002.

[22] A. Zimmermann, D. Reijsbergen, A. Wichmann, and A. Canabal Lavista, "Numerical results for the automated rare event simulation of stochastic Petri nets," in *11th Int. Workshop on Rare Event Simulation (RESIM 2016)*, Eindhoven, Netherlands, 2016, pp. 1–10.

[23] S. Nebel, A. Dieter, P. Müller, and B. Bertsche, "Application of ECSPN to RAMS modeling and analysis of hybrid drive systems," in *Proc. Reliability and Maintainability Symposium (RAMS 2010)*, Jan 2010, pp. 1–6.

[24] A. Zimmermann, *Stochastic Discrete Event Systems*. Springer, Berlin Heidelberg New York, 2007.

[25] M. Villén-Altamirano and J. Villén-Altamirano, "Optimality and robustness of RESTART simulation," in *Proc. 4th Workshop on Rare Event Simulation and Related Combinatorial Optimisation Problems*, Madrid, Spain, Apr. 2002.

[26] M. J. Garvels and D. P. Kroese, "A comparison of RESTART implementations," in *Proc. 1998 Winter Simulation Conference*, 1998.

[27] A. Zimmermann and P. Maciel, "Importance function derivation for RESTART simulations of Petri nets," in *9th Int. Workshop on Rare Event Simulation (RESIM 2012)*, Trondheim, Norway, Jun. 2012, pp. 8–15.

[28] D. Reijsbergen, P.-T. Boer, W. Scheinhardt, and B. Haverkort, "Automated rare event simulation for stochastic Petri nets," in *Quantitative Evaluation of Systems*, ser. Lecture Notes in Computer Science, K. Joshi, M. Siegle, M. Stoelinga, and P. R. D'Argenio, Eds. Springer Berlin Heidelberg, 2013, vol. 8054, pp. 372–388.

[29] A. Zimmermann, "Modeling and evaluation of stochastic Petri nets with TimeNET 4.1," in *Proc. 6th Int. Conf on Performance Evaluation Methodologies and Tools (VALUETOOLS)*. Corse, France: IEEE, 2012, pp. 54–63.

[30] R. German, *Performance Analysis of Communication Systems, Modeling with Non-Markovian Stochastic Petri Nets*. John Wiley and Sons, 2000.

[31] A. Zimmermann, "Reliability modelling and evaluation of dynamic systems with stochastic Petri nets (tutorial)," in *Proc. 7th Int. Conf. on Performance Evaluation Methodologies and Tools (VALUETOOLS 2013)*, Torino, Italy, Dec. 2013.

[32] A. Khajepour, M. Saber Fallah, and A. Goodarzi, *Electric and Hybrid Vehicles: Technologies, Modeling and Control — A Mechatronic Approach*. Wiley, 2014.

[33] M. Trost, S. Nebel, and B. Bertsche, "Modellierungs- und Simulationsumgebung REALIST," in *Proc. Entwicklung komplexer Automatisierungssysteme (EKA 2006)*, E. Schnieder, Ed., Braunschweig, May 2006.

[34] A. Zimmermann and G. Hommel, "Towards modeling and evaluation of ETCS real-time communication and operation," *Journal of Systems and Software*, vol. 77, pp. 47–54, 2005.

[35] A. Zimmermann, S. Jäger, and F. Geyer, "Towards reliability evaluation of AFDX avionic communication systems with rare-event simulation," in *Proc. Probabilistic Safety Assessment & Management Conference 2014 (PSAM 12)*, Honolulu, Hawaii, USA, Jun. 2014, p. 12.