30 Years of Simulation-Based Quantitative Analysis Tools: a Comparison Experiment between Möbius and Uppaal SMC

Davide Basile[1], Maurice H. ter Beek[1], Felicita Di Giandomenico[1], Alessandro Fantechi[1,2], Stefania Gnesi[1], and Giorgio O. Spagnolo[1]

[1] ISTI–CNR, Pisa, Italy
[2] University of Florence, Florence, Italy
{basile,terbeek,digiandomenico,fantechi,gnesi,spagnolo}@isti.cnr.it

Abstract. We provide a brief comparison of the modelling and analysis capabilities of two different formalisms and their associated simulation-based tools, acquired from experimenting with these methods and tools on one specific case study. The case study is a cyber-physical system from an industrial railway project, namely a railroad switch heater, and the quantitative properties concern energy consumption and reliability. We modelled and analysed the case study with stochastic activity networks and Möbius on the one hand and with stochastic hybrid automata and Uppaal SMC on the other hand. We give an overview of the performed experiments and highlight specific features of the two methodologies. This yields some pointers for future research and improvements.

1 Introduction

Industrial critical, cyber-physical systems typically need to satisfy a number of quantitative properties. The formal modelling and efficient analysis of such systems is challenging and has been extensively studied recently. Indeed, simulation-based analysis techniques and tools have been used for decades to perform quantitative analysis, well before the NSF workshop on cyber-physical systems in October 2006 made them fashionable. In particular, stochastic model-based analysis has a longstanding and rich history in Mathematics, well preceding Computer Science as a discipline [29,4]. Statistical Model Checking (SMC) may be traced back to hypothesis testing in the context of probabilistic bisimulation [21,1], but the notion has become popular during the last two decades as a result of Younes's Ph.D. thesis [35,34,22]. Tools that support SMC are more recent [1]. For instance, the first version of Uppaal SMC [17] was released in 2014. The stochastic analysis tool Möbius [15] can be traced back much further, to its predecessors UltraSAN [16,32] and MetaSAN [33]. The latter offer analysis techniques for performability models based on stochastic activity networks (SAN), which are a generalization of stochastic Petri nets [2], which are considered to mark the starting point of cross-fertilization between the fields of performance evaluation and formal verification [4].

In this paper, we continue this cross-fertilization by providing a brief comparison of some of the modelling and analysis capabilities of Möbius and Uppaal SMC, their two different modelling formalisms and their simulation-based quantitative analysis techniques, acquired by experimenting with these methods and tools on one and the same case study. The case study is a cyber-physical system from the railway domain, namely a railroad switch heater, and the quantitative properties concern energy consumption and reliability. This case study comes from our industrial partners in STINGRAY (SmarT station INtelliGent RAilwaY), a project funded by the Tuscany region, which advocates the study of energy-saving algorithms in the railway domain. The models and analyses with stochastic activity networks (SAN) and Möbius have originally been presented in [5], while those with stochastic hybrid automata (SHA) and Uppaal SMC have originally been presented in [9].

The agenda of compared features ranges from modelling features (e.g. communication primitives and delay distributions) to properties specification (e.g. measures of interest) and experiments and presentation of results (e.g. experiment parameter setup), cf. the leftmost column of Table 1 in Section 5. While most of the findings of our comparison are likely well known by the communities around Möbius and Uppaal SMC, this might be less so for someone who is facing her first attempt at modelling a real-time system with the aim of performing quantitative analyses. Our comparison can help such user evaluate which method and tool better fits her specific needs, or at least make her aware of possible limitations and specificities of the chosen methodology. Furthermore, we conclude our comparison by providing some possible pointers to future research and improvements, from the point of view of usability, for both methodologies.

Finally, our approach to model and analyse one and the same case study with two different formalisms and their associated tools also responds to the call for formal methods diversity in the railway sector as put forward in [28,27]. This call, inspired by code/design diversity [24], is based on the assumption that the application of different, non-certified analysis tools on a replication of the same design may increase confidence in the correctness of the analysis results. We believe this to be a useful concept.

Outline After this Introduction, we provide a short description of the case study and its context in Section 2. In Section 3, we briefly describe the two tools, followed by a description of the models and experiments that our comparison is based on in Section 4. The main contribution of our paper is presented in Section 5, where we present a detailed comparison of a number of specific features of the two methodologies, concluded by some pointers to possible improvements for the future. Section 6, finally, wraps up the paper.

2 Context of the Case Study

In this section, we provide a brief description of the case study and project where it originates from. Traditionally, railway stations have a private energy distribution and communication system. The main reasons for this are to ensure uninterrupted power supply and security, but this isolation has two main drawbacks. First, it prohibits integration with ‘smart cities’, in which, ideally, information between different transport systems (i.e. bike sharing, car sharing, urban transport, etc.) is synergically exploited. Second, the station system fails to benefit from modern energy-saving techniques.

The project STINGRAY (SmarT station INtelliGent RAilwaY), funded by the Tuscany region, aims to enhance the integration of railway stations into smart cities of the future as well as to study advanced energy-saving techniques. To this aim, the design and development of a station communication infrastructure is studied, integrating powerline and wireless technologies. Powerlines are utilised to enable a more efficient management of machinery and energetic resources. The goals of the project are:

- to realise a LAN over the station plants using power line and wireless technologies;
- to allow the control and monitoring of station equipment via Supervisory Control And Data Acquisition (SCADA), and in particular railroad switch heaters as studied in this paper;
- to create value-added services for both customers and railway staff, such as connectivity, monitoring fault prediction service (FPS), video surveillance, environmental surveying and integration and access to so-called smart city infomobility services; in particular the energy management service (EMS) is addressed;
- to optimise existing strategies for managing energy consumption within the station, to avoid wasting energy.

The case studies of STINGRAY provided by the industrial partners from the railway domain are station lighting and the heating of the railroad switches in ice conditions (cf. Fig. 1). In this paper, we address the latter case study.

Railroad switch heaters assure correct working of switches in case of ice and snow through a central control unit in charge of managing policies of energy consumption while satisfying reliability constraints. Although apparently a rather focused system, with restricted functionalities, it represents very well the peculiarities of a cyber-physical system: physical components (the heater), cyber components (the heating policies and the related coordinator), stochastic aspects (failure events and weather forecasts), and logical/physical dependencies.

3 Description of the Tools

Before providing the models, we briefly describe Möbius and Uppaal SMC.

3.1 Möbius

Möbius [15] offers a distributed discrete-event simulator, and, for Markovian models, explicit state-space generators and numerical solution algorithms. It is

Fig. 1. Gas heating keeping a railroad switch free from snow and ice (Di Fabian Grunder (FabiBerg) – Opera propria, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=641923)

possible to analyse both transient and steady-state reward models. Möbius supports different formalisms, among which the aforementioned SAN, and models specified in different formalisms can be composed in different ways. Reward models are used to define the measures under analysis. A SAN is composed of the following primitives: *places*, *activities*, *input gates*, and *output gates*. Places and activities have the same interpretation as places and transitions in Petri nets [30]. Input gates control the enabling conditions of an activity and define the change of marking when an activity completes. Output gates define the change of marking upon completion of the activity. Activities can be of two types: *instantaneous* or *timed*. Instantaneous activities complete once the enabling conditions are satisfied. Timed activities take an amount of time to complete, following a temporal stochastic distribution function which can be, e.g., exponential or deterministic. Cases are associated to activities, and are used to represent probabilistic uncertainty about the action taken upon completion of the activity. Primitives of the SAN models are defined using C++ code.

3.2 Uppaal SMC

Statistical Model Checking (SMC) concerns running a sufficient number of (probabilistically distributed) simulations of a system model to obtain statistical evidence (with a predefined level of statistical confidence) of the quantitative properties to be checked [1,22]. UPPAAL SMC [17] is an extension of UPPAAL [11], a well-known toolbox for the verification of real-time systems modelled by (extended) timed automata. Timed automata are finite-state automata enhanced with real-time modelling through *clock* variables; their stochastic extension replaces non-determinism with probabilistic choice and time delays with probability distributions (uniform for bounded time and exponential for unbounded time). These automata may communicate via (broadcast) channels and shared variables. The resulting stochastic hybrid automata (SHA) form the input models of Uppaal SMC. Uppaal SMC allows to check (quantitative) properties over simulation runs of an Uppaal SMC model (i.e. a network of SHA). These properties must be expressed in a dialect [12] of the Metric Interval Temporal Logic (MITL) [3].

4 Models and Experiments

To contextualize the comparison between the two methodologies, in this section we briefly describe the models and experiments performed in [5,9].

Although the modeling studies on the railroad switch heating system have since been further extended to focus on other aspects (mainly, to account for more sophisticated weather dynamics and representation, e.g. in [14]), these only exploited the SAN formalism and Möbius. Hence for our comparison experiment the two works mentioned in the beginning of this section are the most suitable.

4.1 Modelling Approaches

The aforementioned energy-saving policies for railroad switch heaters are based on dynamic power management, according to which energy is turned on and off based on predefined temperature thresholds. Moreover, the system can be constrained to not exceed a given maximum amount of power. This is especially useful in case of degraded operational modes which forbid to exceed a certain amount of power. In particular, once the system temperature falls below a temperature warning threshold (T_{wa}), the heating needs to be activated, otherwise the associated switch fails. Once the temperature rises and reaches the working threshold (T_{wo}), the heating system can be safely turned off.

The models (in both tools) are parameterized based on these two temperature thresholds T_{wa} and T_{wo}, and on NH_{max}, which is the maximum power that the system can provide at every instant of time, expressed as the percentage of heaters that can be turned on at the same time.

The continuous physical behaviour concerning the increment and decrement of the temperature of the railroad track when the heater is turned on or off, respectively, is modelled by an ordinary differential equation (ODE) representing the balance of energy.

When the temperature of the railroad track is below the freezing threshold (i.e. 0°C in the performed experiments), a switch may experience a failure. In this case, the time-to-failure is modelled with an exponential distribution with fixed rate, which is based on the temperature of the railroad track. This rate is an input parameter.

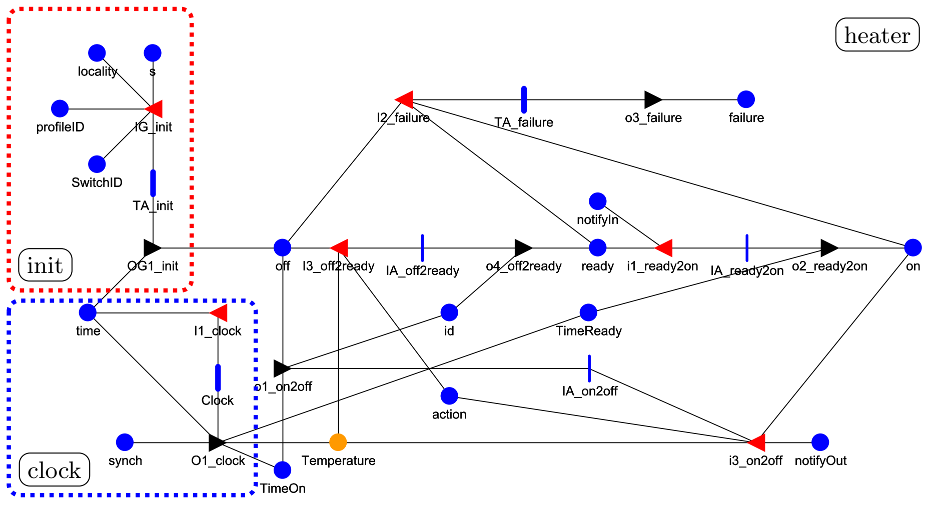

Fig. 2. The SAN model *RailRoadSwitchHeater* from [5]

To model the external weather conditions, the model takes as input data structures containing profiles of average temperatures in those days for which the analysis is relevant (e.g. winter days). Different daily weather profiles retrieved from the Internet are used in the performed experiments. The time window under analysis is divided into intervals to which an average reference temperature is assigned.

The two main logical components describing the discrete cyber part of the analysed system are the *heater* and the *central coordinator*. The overall model is then composed of n heaters and the coordinator. The heater model implements the policy for activating and deactivating the heating phase. The central coordinator manages the activation and deactivation of each heater, by interacting with the network of heaters, to notify the activation or deactivation, respectively, of a heater, according to a specific communication protocol designed by the authors.

SAN model We first describe the SAN model of the railroad switch heating system, built with the functionalities provided by Möbius.

The main SAN model concerning the railroad switch heater is depicted in Fig. 2, reproduced from [5]. It is partitioned into three logical components: the *init* subnet, the *clock* subnet, and the *heater* subnet.

The *init* subnet initialises the data structures used by the SAN model. The *clock* subnet models the evolution of time (during one day in our analysis) and it is used to update the environment temperature and the temperature of the railroad track. In [5], we considered as unit of time one hour. The activity `Clock` has a deterministic distribution of time (non-Markovian) and completes each hour. When `Clock` completes, the place `Temperature` is updated: if the heater

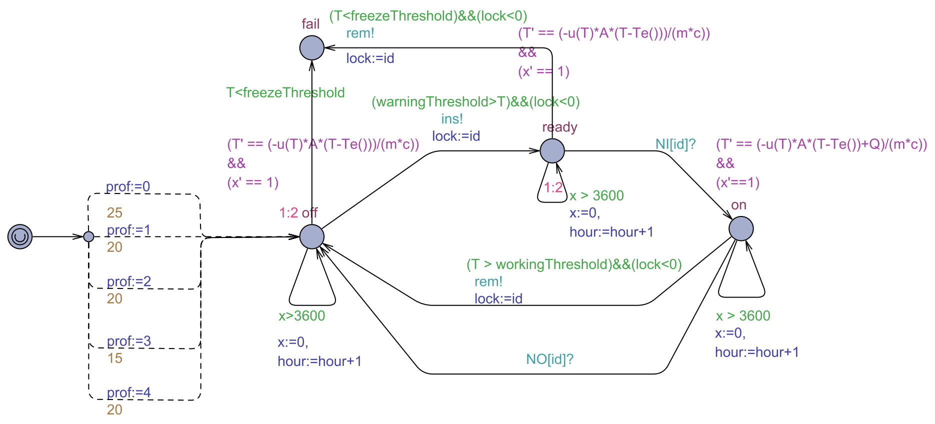

Fig. 3. SHA H from [9], modelling an instance of a railroad switch heater

is turned on then the temperature increases, otherwise the temperature will be updated according to the temperature of the environment. Indeed, the time-step has been discretized to account for the temperature profile windows.

The *heater* subnet represents the status of the railroad switch heater. The *heater* subnet interacts with a SAN model *Coordinator* (not depicted here) through places that are shared among all the replicas of the heater model and the coordinator model.

The function representing the heating exchange is defined in C++, and it is called by the output gate `O1_clock` shown in Fig. 2 to update the temperature of the railroad each interval of time t. The activity `TA_failure` models the failure of a heater. It has an exponential distribution of time based on the temperature of the railroad track: the more the temperature is below the freezing threshold the more likely the activity will fire, according to the rate of the distribution which is an input parameter of the model.

The SAN model *Coordinator* represents the central management unit and it interacts with all heaters in the network by activating, deactivating, or moving them into a waiting state.

SHA model Next we describe the SHA model of the railroad switch heating system, built with the functionalities provided by Uppaal SMC. SHA allow to capture discrete, continuous, and stochastic aspects in a single framework. We briefly outline the formalisation of the system of (remotely controlled) railroad switch heaters as a product of SHA.

The ODE is expressed in the SHA model `H` in Fig. 3, where the temperature T is a *continuous clock* and the flow function F (i.e. the ODE) is similar in different states. Indeed, when `H` is in state `on`, F adds the term `Q` (i.e. power), which does not occur in states `off` and `ready`.

The two main logical components describing the discrete cyber part of the analysed system are the *heater* `H` and the *central coordinator* `K`, depicted in

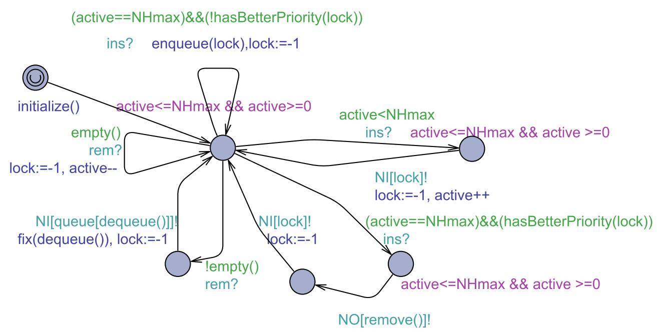

Fig. 4. SHA `K` from [9], modelling the coordinator

Figs. 3 and 4, respectively, both reproduced from [9]. The network composed of n heaters and the coordinator is realised by the product of `K` and the replicas of the SHA $\mathtt{H}_{\mathtt{id}}$, $\mathtt{id} \in 1, \ldots, n$, where each heater is uniquely identified by its `id`, i.e. $(\bigotimes_{\mathtt{id} \in 1,\ldots,n} \mathtt{H}_{\mathtt{id}}) \otimes \mathtt{K}$.

The SHA heater model depicted in Fig. 3 implements the policy for activating and deactivating the heating phase, similarly to the SAN heater model depicted in Fig. 2. In particular, the dotted transitions are urgent (i.e. instantaneous) probabilistic transitions used for selecting one of the available weather profiles. The main states are `on`, `off`, `ready`, and `fail`, which correspond to the places of the SAN model *RailRoadSwitchHeater*. Note that each state has an inner cycle modelling the decrease and increase of the internal temperature according to the flow function, and that both incoming transitions to state `fail` have an exponential distribution of time, whose rates are input parameters to fine tune the model. During a simulation, the current time is stored in the clock `x` and a variable `hour` stores the current hour. The function `Te`(), used in the flow function of `T`, selects the actual external temperature based on the current hour, and it is implemented in Uppaal.

The SHA model depicted in Fig. 4 implements the coordinator model. Its behaviour is similar to that of the SAN model *Coordinator* mentioned above. The queue of pending heaters is modelled with the array `queue`[ ] of length equal to NH_{max}, and the functions `enqueue(int id)` and `dequeue`() are used for inserting and removing elements, while `empty`() returns true if the queue of pending heaters is empty.

The coordinator sends messages to the network of heaters through two arrays of channels, `NI`[`id`] and `NO`[`id`], both indexed by the identifiers of the heaters, to notify the activation and deactivation, respectively, of a heater. Note that Uppaal SMC only allows broadcast channels, hence an array of channels has been adopted in order to implement one-to-one communications. Following the de facto standard notation in component-based systems, sending a message through a channel `a` is denoted as `a`!, while reading is denoted as `a`?. Upon reception of the notification NI[id]?, the heater with identifier `id` switches from state `ready` to state `on`.

The heaters communicate to the coordinator their transition from `off` to `ready` through the channel `ins`, asking to be activated, and their transition from `on` to `off` through the channel `rem`; both channels are many-to-one. All channels are *urgent*: no delay will occur in case a synchronisation is available.

While the coordinator is in a `busy` state, a shared variable `lock` is used as a semaphore to prevent a heater from sending messages that cannot be elaborated, and it is used by the heaters for communicating their identifiers to the coordinator.

4.2 Quantitative Analyses

The conducted analyses focussed on reliability as well as energy consumption indicators. More precisely, the two measures of interest concerning energy consumption and reliability of the system under analysis were defined as follows.

1. The time (in hours) a generic heater is activated in a specific time interval. By multiplying such measurement for the power consumed (kilowatt per hour), it is possible to derive the energy consumed by the system.
2. The probability that a generic switch fails (becomes frozen). Reliability is computed as the probability that no failure occurs in the interval of time under analysis.

In Möbius, reward structures were used for evaluating the measures of interest, while in Uppaal SMC they were defined as formulae in the aforementioned MITL [3], which are enriched with quantification operators on the replicated models and expected values.

First consider the SAN model. The first measure of interest was computed as the sum of the time that each heater model spends in markings encoding its operative state, that is the time that each heater is activated. The second measure of interest, the probability of failure, instead was computed as the probability that there is one token in the place encoding a failure state in the heater model at the end of the experiment.

Next consider the SHA model. In Uppaal SMC, a discrete clock `energy` was used to count the hours each heater is activated. For the first measure of interest, the energy consumption, the number of hours in which the heaters are active was estimated as the formula:

$$\mathtt{E}[<= 24; 10000]\ (\mathtt{max} : \sum_{\mathtt{i:id_t}} \mathtt{H_i.energy})$$

In this formula, `E` stands for the expected value, 24 is the considered interval of time (24h) and 10000 is the number of simulations executed by the tool. The overall energy consumption is the sum for all $\mathtt{H_i}$ of all clocks `energy`.

The second measure of interest, the probability of failure, was instead estimated by Uppaal SMC through the formula:

$$\mathbb{P}(\Diamond_{\mathtt{h}\leq 24}\exists(\mathtt{i} : \mathtt{id_t})(\mathtt{H_i.fail}))$$

This formula evaluates the probability that in the interval $[t, t+l]$ (24h) there exists at least a switch `Hi` in the network which has failed, i.e. `Hi` is in state `fail`.

Experiments conducted on a system of 10 switches, grouped according to their priority, confirmed that both modelling and analysis methodologies, i.e. SAN and Möbius on the one hand and SHA and Uppaal SMC on the other hand, are suitable to address the analyses. Results were aligned in spite of some differences in the models, thus also serving as mutual cross-validation, as advocated by formal methods diversity (cf. Introduction).

As expected, the analyses confirmed how energy consumption and reliability are contrasting requirements: by reducing the energy consumption the overall system reliability decreases. However, experiments made it possible to find a parameter setup that represents the best compromise between these two measures. Of course, the nature of the adopted approaches did generate differences in the evaluation experience, as discussed in the next section.

5 Comparison

In this section, we present the main contribution of our paper, in the form of a number of considerations subjective to our experiences with applying the two methodologies to the above mentioned case study. Our aim is merely to highlight some specific features that the two methodologies offer, to be used by potential modellers in deciding which one better suits their need for the particular case study at hand. In addition, we underline that the comparison carried out touches only those functionalities of the two modeling and evaluation environments that were involved given the needs of the case study under analysis. Therefore, we cannot claim that we exhaustively considered all features, and consequently our goal is by no means to pronounce a definitive verdict concerning the methodologies' suitability, let alone quality. Generally speaking, we note that Möbius is a mature tool that has been widely adopted in the evaluation of performance and dependability aspects of real-world systems, while Uppaal is a mature tool oriented to the quantitative verification of properties of real-time systems of which Uppaal SMC is a recent, as yet less mature extension.

Our comparison should be seen in the light of a recent study, reported in [10], of the outcomes of three questionnaires on the adoption of formal methods and tools in the railway domain, which were performed within three different projects of the EU Shift2Rail innovation programme (cf. https://shift2rail.org/). As part of an analysis of the respondents' expectations on tools, the paper reports that the most relevant functionalities are formal verification and support for formal modelling, followed by traceability, simulation, test and code generation. Instead, the most relevant quality features are related to the maturity, usability, and learnability of the tools.

Our comparison addresses the following three groups of features (summarised in the leftmost column of Table 1 below) in the next three sections.

Modelling Features: these concern the composition of, and interactions between, different models (i.e. *heterogeneous formalisms, replicated models,* *dynamic process instantiation, communication primitives*) and the ability towards modelling hybrid and stochastic systems (i.e. *delay distributions, hybrid variables*);

Properties Specification: these concern the definition of measures of interest (i.e. *measures of interest*) and the ability to verify properties of the defined models (i.e. *property verification*);

Experiments and Presentation of Results: these concern the setup and execution of experiments, as well as data collection and plotting the results (i.e. *experiment parameter setup*).

We note that, for this comparison experiment that is focused on usability and expressiveness, we specifically consider whether features are primitively supported by the tool. Of course, for features that are not built-in, both tools may rely on external software packages or libraries for accessing extended functionalities at the cost of an extra modelling effort. In the end, we provide some pointers to future research and improvements for both methodologies.

5.1 Modelling Features

Systems under analysis are often composed of different types of components, and in general more components of the same type may be involved, as is the case for the railroad switch heater system of our case study. Möbius allows to develop a composite overall model where individual models can be replicated, joined, and defined in different formalisms (e.g. Petri nets [30,31], PEPA [19], Fault trees [23], etc.). However, replication operators treat models as anonymous, so in case non-anonymous instantiations are required, as in our case study, a specific mechanism to assign a unique identifier to each replica model needs to be added to the model under development. Anticipating the discussion, we mention that the replication operator in Möbius has recently been enhanced in efficiency [26], thus alleviating to some extent the additional computational overhead required by the mechanism to implement non-anonymity. Moreover, efficient solutions to non-anonymous model replication have been also proposed (cf., e.g., [25,13]), which resort to a script on top of primitive facilities provided by Möbius.

Similarly, Uppaal offers template models that can be replicated, but each replica has its own built-in identifier. Identifiers can be used for quantifying formulae, as we did for the measures of interest in our case study. It is also possible to dynamically create new processes during a simulation, through a *fork* primitive. Several instances of models can be joined through the composition operator. Accordingly, when different formalisms are necessary for designing the system under analysis and the replicas are anonymous, Möbius is more adequate. If the replicas are instead non-anonymous, Möbius requires to distinguish them through ad-hoc networks that result in a larger state space. Moreover, if the system to be modelled comprehends the dynamic generation of new processes, then this feature is primitively available in Uppaal, making it very suitable.

Concerning the interaction between different instances of models, in the SAN models defined in Möbius communication is implemented through shared places (i.e. places where different networks can read/write). Indeed, through tokens in different places it is possible to codify the identifiers of the interacting parties and the messages sent. The SHA models defined in Uppaal SMC are endowed with primitives for I/O communication, allowing to describe interactions among entities in a high-level language. When modelling communication-based systems, SHA models thus offer both I/O primitives at message level and shared variables, while SAN models interact through shared places (acting as shared variables). We note in passing that shared places/variables are part of the state space, while synchronous messages may reduce the state space by avoiding interleavings (e.g. places `notifyIn` and `notifyOut` in Fig. 2 are rendered as communication channels `NI` and `NO` in Fig. 3).

Concerning modeling stochastic and hybrid Systems, both SAN and SHA models are capable of describing probabilistic transitions and stochastic delays. Through SAN models it is possible to describe Markovian and non-Markovian models with several probability distributions for delays in firing a transition, whereas only uniform and exponential distributions for delays are available for SHA models. Therefore, in general, SAN models allow for a more accurate representation of physical phenomena. Both formalisms can model instantaneous transitions, which are called instantaneous activities in SAN models and urgent transitions in SHA models.

SHA models allow to describe discrete and hybrid clocks for updating values according to given ODE, whereas SAN models do not provide a built-in solver of ODE. Instead, the equations have to be solved and implemented in, for example, C++ functions or via calls to external solvers. The hybrid clocks are stored in Uppaal SMC through double precision types, while Möbius provides extended places for storing high precision values.

Hence, Uppaal SMC deals with hybrid systems by primitively supporting ODE. Möbius, on the other hand, primitively allows to model several stochastic distributions in SAN models.

5.2 Properties Specification

Concerning definition of measures of interest, in Möbius, measures of interest (performance variables) on the composed model are defined through reward models. A reward model defines the data that needs to be collected from the model (using C++ code), through analytic solvers or simulations. Rate rewards on the measures of interest specify whether the reward is collected based on the marking of the SAN models or the firing of activities, and if it is collected at a specific instant of time, over an interval of time, over a time-averaged interval of time, or after the system reaches a steady state.

In Uppaal SMC, measures of interest are introduced by means of formulae in a weighted extension of MITL [12]. The available evaluation methods are probability estimation, hypothesis testing, and probability comparison. It is also possible to perform simulations to monitor the values of interest. The possibility of expressing measures of interest as formulae in a temporal logic has the advantage that a precise formal semantics endows those measures. Moreover, it is possible to define fine-grained properties directly through the available temporal operators. For example, the formula $\mathbb{P}(\Diamond_{[0,24]}\exists(\mathtt{i} : \mathtt{id_t})(\Box_{[0,2]}\mathtt{H_i}.\mathtt{on}))$ evaluates the probability that there exists, in the interval of 24 hours, a component $\mathtt{H_i}$ (where *i* is its index) in state *on*, for at least 3 consecutive time units.

Uppaal offers high-level expressions for the formal definition of varieties of indicators to be analysed, while Möbius typically requires to enrich the model to properly account for sophisticated properties, such as the one described by the temporal logic formula above (e.g., by adding ad-hoc places and transitions to code the property to be analysed, thus resulting in a more complex model). Continuing the example, an extra place should be added to Fig. 2 with a token being added once the heater is `on` for more than 3 time units, and the reward model should be based upon this extra place.

Concerning the verification of the models and performances, Uppaal provides the possibility of verifying properties such as for example the absence of deadlocks. It is also possible to perform trace analysis and simulation of the models for debugging purposes, and in case a property is violated the tool reports the trace which violates it as counterexample.

Möbius does not provide any built-in verification of properties expressed in some kind of logic. The property must be encoded in a Markov Reward Model, thus requiring more effort from the point of view of the user. However, it is possible to perform LTL model checking on traces obtained from the logs of the simulations in Möbius by means of external prototypical tools, such as for example Traviando [20].

Summing up, Uppaal supports model checking of temporal logic formulae, while Möbius offers a more prototypical trace-based analysis.

Finally, in relation to the specific experiments carried out in [5,9], Möbius showed better performances than Uppaal SMC. This might be due to the fact that Uppaal SMC solves the defined ODE during simulation and the number of simulations is fixed. No general conclusions can be drawn from two experiments.

5.3 Experiments and Presentation of Results

The experiments in Möbius can be organised in batches, called studies. Each study contains the parameter setup of the experiment (such as temperature thresholds and energy available in our case), which can then be executed in series or in parallel. This feature enhances efficiency, by allowing to perform all required experiments in background. In Uppaal SMC, instead, the parameters must be instantiated manually for each experiment to be evaluated.

The results of the experiments performed by Möbius are stored into tabular data, ready to be analysed and plotted through a database (PostgreSQL) or external tools. Uppaal provides built-in graphic visualisers of, for example, the density and cumulative distribution of the evaluated property.

In the specific case study presented in this paper, Möbius proved effective in dealing with several parameter setups and a large amount of resulting data concerning the experiments. From the viewpoint of presentation of the results, Uppaal automatically generates the visualisation of results while Möbius requires a pre-processing phase.

5.4 Discussion

The comparison performed in this section so far is summarised in Table 1, which reflects one of the main contributions of this paper. In the remainder of this section, we highlight some directions for future developments for both Möbius and Uppaal SMC to address some of the points discussed so far.

Table 1. Comparison between SAN + Möbius and SHA + Uppaal SMC

Features	SAN + Möbius	SHA + Uppaal SMC
Measures of interest	Reward Models	MITL formulae
Experiments parameter setup	Batches	Single
Replicated models	Anonymous	Distinguished
Dynamic process instantiation	Not available	Available
Heterogeneous formalisms	Available (SAN, PEPA, etc.)	Not available (SHA)
Communication primitives	Shared places	Channels
Delay distributions	Various distributions	Exponential, Uniform
Hybrid variables	No primitive support	ODE solver available
Property verification	Not available	Temporal logics

Concerning SAN and Möbius, improvement of the anonymous replication aspect appears to be a major advancement to pursue. Actually, this is already ongoing activity (involving a subset of the authors), aiming at implementing the principles at the basis of the replication mechanism defined in [13] as a native Möbius operator.

A further interesting extension of Möbius, from the point of view of usability, would be the automatic generation of plot graphs from predefined measures of interest. Indeed, although Möbius is conceived as a meta-tool to be coupled with a variety of other tools (including visual tools and other functionalities), offering an internal visualization facility would be certainly appreciated by those users that prefer to have all they need within the same working environment. Motivated by the same reasoning, also a primitive support for ODE solving would be a step towards making easier the modeling effort of cyber-physical systems, without requiring the knowledge of software libraries tailored to specific needs.

Moving to SHA and Uppaal SMC, we note that the tool lacks the possibility of organising experiments in batches, where each batch has a specific parameter setup. In the current version, this has to be done manually or by means of external scripts. An interesting facility from the point of view of usability would be to equip the tool with the possibility to automatically execute batches of experiments (and collect the results). The possibility to primitively express other distribution delays (different from exponential distributions) would increase the expressiveness of the SHA formalism. For example, the SAN model presented previously discretises the time-steps by simply having a deterministic distribution delay that fires each specific time unit. We note in passing that such behaviour (deterministic time) is typical of many real-time specifications, i.e. interacting periodically with a fixed period. This behaviour can be obtained in Uppaal by an encoding of an invariant on a state of the form $x \leq t$ and an outgoing transition from that state of the form $x \geq t$. This ensures that the transition is fired exactly at time t. Note that in Uppaal invariants are defined for each state individually. Making such behaviour primitively expressible in the tool would improve its usability, as well as the readability of the models. Furthermore, it would typically reduce the state space (in the above example, we would need to instantiate a clock x and a parameter t).

Finally, we envisage that a formal mapping from SHA to SAN models would pave the way for automatic replicas of analyses, thus increasing the confidence in results and the soundness of the tools' implementations. In [7], a subset of the authors already provided a formal translation from contract automata [6] to SAN. Contract automata are similar to the SHA formalism used in Uppaal, and a timed extension also exists [8]. This former translation could be extended to deal with stochastic delays (to be encoded in SAN activities) and real-time clocks.

6 Conclusion

We have compared the modelling and analysis capabilities offered by SAN and Möbius with those offered by SHA and Uppaal SMC. This comparison experiment is based on modelling and analysing a single, small cyber-physical system from an industrial railway project (cf. [18] for a judgement study involving Uppaal SMC and 8 other tools). We have provided an overview of the performed experiments and based on those we have highlighted some specific features of the two methodologies. This has resulted in a few pointers for future research and improvements for both, to be considered during the next 30 years.

Acknowledgements Supported by POR FESR 2014-2020 project STINGRAY (SmarT station INtelliGent RAilwaY) and MIUR PRIN 2017FTXR7S project IT MaTTerS (Methods and Tools for Trustworthy Smart Systems).

References

1. Agha, G., Palmskog, K.: A Survey of Statistical Model Checking. ACM Trans. Model. Comput. Simul. **28**(1), 6:1–6:39 (2018). doi:10.1145/3158668
2. Ajmone Marsan, M., Bobbio, A., Donatelli, S.: Petri nets in performance analysis: An introduction. In: Reisig and Rozenberg [30], pp. 211–256. doi:10.1007/3-540-65306-6_17
3. Alur, R., Feder, T., Henzinger, T.A.: The Benefits of Relaxing Punctuality. J. ACM **43**(1), 116–146 (1996). doi:10.1145/227595.227602

4. Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.P.: Performance Evaluation and Model Checking Join Forces. Commun. ACM **53**(9), 76–85 (September 2010). doi:10.1145/1810891.1810912
5. Basile, D., Chiaradonna, S., Di Giandomenico, F., Gnesi, S.: A stochastic model-based approach to analyse reliable energy-saving rail road switch heating systems. J. Rail Transp. Plan. Manag. **6**(2), 163–181 (2016). doi:10.1016/j.jrtpm.2016.03.003
6. Basile, D., Degano, P., Ferrari, G.L.: Automata for specifying and orchestrating service contracts. Log. Meth. Comp. Sci. **12**(4) (2016). doi:10.2168/LMCS-12(4:6)2016
7. Basile, D., Di Giandomenico, F., Gnesi, S.: A Refinement Approach to Analyse Critical Cyber-Physical Systems. In: Cerone, A., Roveri, M. (eds.) SEFM. LNCS, vol. 10729, pp. 267–283. Springer (2017). doi:10.1007/978-3-319-74781-1_19
8. Basile, D., ter Beek, M.H., Legay, A.: Timed service contract automata. Innovations Syst. Softw. Eng. **16**(2), 199–214 (2020). doi:10.1007/s11334-019-00353-3
9. Basile, D., Di Giandomenico, F., Gnesi, S.: Statistical Model Checking of an Energy-Saving Cyber-Physical System in the Railway Domain. In: Proceedings of the 32nd Symposium on Applied Computing (SAC). pp. 1356–1363. ACM (2017). doi:10.1145/3019612.3019824
10. ter Beek, M.H., Borälv, A., Fantechi, A., Ferrari, A., Gnesi, S., Löfving, C., Mazzanti, F.: Adopting Formal Methods in an Industrial Setting: The Railways Case. In: ter Beek, M.H., McIver, A., Oliveira, J.N. (eds.) FM. LNCS, vol. 11800, pp. 762–772. Springer (2019). doi:10.1007/978-3-030-30942-8_46
11. Behrmann, G., David, A., Larsen, K.G., Håkansson, J., Pettersson, P., Yi, W., Hendriks, M.: UPPAAL 4.0. In: QEST. pp. 125–126. IEEE (2006). doi:10.1109/QEST.2006.59
12. Bulychev, P., David, A., Larsen, K.G., Legay, A., Li, G., Poulsen, D.B.: Rewrite-Based Statistical Model Checking of WMTL. In: Qadeer, S., Tasiran, S. (eds.) RV. LNCS, vol. 7687, pp. 260–275. Springer (2013). doi:10.1007/978-3-642-35632-2_25
13. Chiaradonna, S., Di Giandomenico, F., Masetti, G.: A Stochastic Modeling Approach for an Efficient Dependability Evaluation of Large Systems with Non-anonymous Interconnected Components. In: Proceedings of the 28th International Symposium on Software Reliability Engineering (ISSRE). pp. 46–55. IEEE (2017). doi:10.1109/ISSRE.2017.17
14. Chiaradonna, S., Giandomenico, F.D., Masetti, G., Basile, D.: A Refined Framework for Model-Based Assessment of Energy Consumption in the Railway Sector. In: ter Beek, M.H., Fantechi, A., Semini, L. (eds.) From Software Engineering to Formal Methods and Tools, and Back. LNCS, vol. 11865, pp. 481–501. Springer (2019). doi:10.1007/978-3-030-30985-5_28
15. Clark, G., Courtney, T., Daly, D., Deavours, D., Derisavi, S., Doyle, J.M., Sanders, W.H., Webster, P.: The Möbius Modeling Tool. In: Proceedings of the 9th International Workshop on Petri Nets and Performance Models (PNPM). pp. 241–250. IEEE (2001). doi:10.1109/PNPM.2001.953373
16. Couvillion, J.A., Freire, R.S., Johnson, R., Obal II, W.D., Qureshi, M.A., Rai, M., Sanders, W.H., Tvedt, J.E.: Performability Modeling with UltraSAN. IEEE Softw. **8**(5), 69–80 (1991). doi:10.1109/52.84218
17. David, A., Larsen, K.G., Legay, A., Mikučionis, M., Poulsen, D.B.: Uppaal SMC tutorial. Int. J. Softw. Tools Technol. Transf. **17**(4), 397–415 (2015). doi:10.1007/s10009-014-0361-y
18. Ferrari, A., Mazzanti, F., Basile, D., ter Beek, M.H., Fantechi, A.: Comparing Formal Tools for System Design: a Judgment Study. In: Proceedings of the 42nd

International Conference on Software Engineering (ICSE). pp. 62–74. ACM (2020). doi:10.1145/3377811.3380373

19. Hillston, J.: A Compositional Approach to Performance Modelling. Cambridge University Press (1996). doi:10.1017/CBO9780511569951
20. Kemper, P., Tepper, C.: Traviando - Debugging Simulation Traces with Message Sequence Charts. In: QEST. pp. 135–136. IEEE (2006). doi:10.1109/QEST.2006.58
21. Larsen, K.G., Skou, A.: Bisimulation through Probabilistic Testing. Inf. Comput. **94**(1), 1–28 (1991). doi:10.1016/0890-5401(91)90030-6
22. Legay, A., Lukina, A., Traonouez, L., Yang, J., Smolka, S.A., Grosu, R.: Statistical Model Checking. In: Steffen, B., Woeginger, G.J. (eds.) Computing and Software Science: State of the Art and Perspectives, LNCS, vol. 10000, pp. 478–504. Springer (2019). doi:10.1007/978-3-319-91908-9_23
23. Limnios, N.: Fault Trees. ISTE (2007). doi:10.1002/9780470612484
24. Littlewood, B., Popov, P., Strigini, L.: Modeling Software Design Diversity: A Review. ACM Comput. Surv. **33**(2), 177–208 (2001). doi:10.1145/384192.384195
25. Masetti, G., Chiaradonna, S., Di Giandomenico, F.: Model-Based Simulation in Möbius: An Efficient Approach Targeting Loosely Interconnected Components. In: Reinecke, P., Di Marco, A. (eds.) EPEW. LNCS, vol. 10497, pp. 184–198. Springer (2017). doi:10.1007/978-3-319-66583-2_12
26. Masetti, G., Chiaradonna, S., Di Giandomenico, F., Feddersen, B., Sanders, W.H.: An Efficient Strategy for Model Composition in the Möbius Modeling Environment. In: Proceedings of the 14th European Dependable Computing Conference (EDCC). pp. 116–119 (2018). doi:10.1109/EDCC.2018.00029
27. Mazzanti, F., Ferrari, A.: Ten Diverse Formal Models for a CBTC Automatic Train Supervision System. In: Gallagher, J.P., van Glabbeek, R., Serwe, W. (eds.) MARS/VPT. EPTCS, vol. 268, pp. 104–149 (2018). doi:10.4204/EPTCS.268.4
28. Mazzanti, F., Ferrari, A., Spagnolo, G.O.: Towards formal methods diversity in railways: an experience report with seven frameworks. Int. J. Softw. Tools Technol. Transf. **20**(3), 263–288 (2018). doi:10.1007/s10009-018-0488-3
29. Pinsky, M.A., Karlin, S.: An Introduction to Stochastic Modeling. Academic Press, 4th edn. (2011). doi:10.1016/C2009-1-61171-0
30. Reisig, W., Rozenberg, G. (eds.): Lectures on Petri Nets I: Basic Models, LNCS, vol. 1491. Springer (1998). doi:10.1007/3-540-65306-6
31. Reisig, W., Rozenberg, G. (eds.): Lectures on Petri Nets II: Applications, LNCS, vol. 1492. Springer (1998). doi:10.1007/3-540-65307-4
32. Sanders, W., Obal II, W., Qureshi, M., Widjanarko, F.: The UltraSAN modeling environment. Perform. Eval. **24**(1), 89–115 (1995). doi:10.1016/0166-5316(95)00012-M
33. Sanders, W.H., Meyer, J.F.: METASAN: A Performability Evaluation Tool Based on Stochastic Acitivity Networks. In: Proceedings of the 1986 Fall Joint Computer Conference. pp. 807–816. IEEE (1986)
34. Sen, K., Viswanathan, M., Agha, G.: Statistical Model Checking of Black-Box Probabilistic Systems. In: Alur, R., Peled, D.A. (eds.) CAV. LNCS, vol. 3114, pp. 202–215. Springer (2004). doi:10.1007/978-3-540-27813-9_16
35. Younes, H.L.S.: Verification and Planning for Stochastic Processes with Asynchronous Events. Ph.D. thesis, Carnegie Mellon University (January 2005), http://reports-archive.adm.cs.cmu.edu/anon/2005/CMU-CS-05-105.pdf