MODEST – A UNIFIED LANGUAGE FOR QUANTITATIVE MODELS

Arnd Hartmanns

Saarland University – Computer Science, Saarbrücken, Germany

ABSTRACT

MODEST is a behavioural modelling language for stochastic timed systems, which allow the representation of both probabilistic and real-time aspects together with nondeterministic decisions and abstractions. Rooted in process algebra, it has an expressive syntax enriched with features from programming languages, leading to concise models with a clearly defined semantics. A key idea behind MODEST is the single-formalism, multiple-solution approach: A range of analysis options such as discrete-event simulation and different variants of model checking are available for a single MODEST model. This paper gives an introduction to the MODEST language and its underlying semantics, followed by a survey of the current state of analysis approaches and successful applications of MODEST to a diverse range of case studies.

1. INTRODUCTION

Our reliance on complex safety-critical or economically vital systems such as fly-by-wire controllers, networked industrial automation systems or "smart" power grids increases at an ever-accelerating pace. The necessity to study the reliability and performance of these systems is evident. Over the last two decades, significant progress has been made in the area of formal methods to allow the construction of mathematically precise models of such systems and automatically evaluate properties of interest on the models. Classically, model checking has been used to study functional properties related to correctness such as such as "the system will never reach a bad state" (*safety*) or "whenever the traffic light is red, it will become green again in the future" (*liveness*). However, since e.g. correct system implementations may still be unusably slow or energy-consuming, performance requirements need to be considered as well. This need to evaluate both *qualitative* as well as *quantitative* properties fostered the development of integrative approaches that combine probabilities, real-time aspects or costs with formal verification techniques [1], allowing questions such as "what is the expected time until a task is completed" or "what is the minimum probability of success without exceeding battery capacity" to be answered.

Today, quantitative modelling and analysis is supported by a wide range of tools and formalisms such as—to name only a few examples—the CADP [2] toolkit centered around the LOTOS language [3], PRISM [4] and similar tools [5, 6, 7] operating on guarded commands, or UPPAAL [8], which allows the graphical modelling of networks of timed automata (TA, [9]). Notably, most of the mentioned tools, which come from a model-checking background, now also feature dedicated simulation or performance evaluation features [10, 11]. The variety of different languages used by tools in this area, however, is a major obstacle for new users seeking to apply formal methods in their field of work. Most input languages are understood by only a single tool, which in turn is usually dedicated to a particular formalism (such as probabilistic, but not real-time systems) or even a specific analysis method (e.g. steady-state analysis, but not transient properties or bounded model checking).

MODEST [12], on the other hand, is a language designed to be as expressive as possible instead of being restricted to particular analysis approaches. It has a formal semantics in terms of stochastic timed automata (STA), which are a rich formalism that includes nondeterministic and discrete probabilistic choices as in probabilistic automata (PA, [13]), hard real-time behaviour as in TA as well as stochastic sampling and delays according to arbitrary probability distributions. In fact, many well-known and extensively studied models such as Markov chains, PA or TA are special cases of STA, and most are easy to identify on the syntactic level in MODEST. While rooted in process algebra, MODEST borrows syntax and concepts from widely-used programming languages to make the language accessible to programmers and engineers. At the same time, its expressive syntax allows complex models to remain concise and readable.

As an overarching language for models focussing on many different qualitative and quantitative aspects, MODEST was—at the time it was designed—too expressive for a single tool to provide comprehensive analysis and verification support. MODEST has thus been designed with a *single-formalism, multiple-solution* approach in mind: Analysis tools for MODEST focus on restricted subsets of the language (and thus on different submodels of STA) or specific analysis methods, reusing existing tools as backends in order to avoid unnecessary reimplementations. This allows a single model to be analysed with a range of methods. For example, consider the PA subset: A state-of-the-art probabilistic model checker such as PRISM can be used as backend to perform a fully probabilistic analysis. For functional properties, though, it often suffices to consider an overapproximation of the PA

This work has been supported by the DFG as part of SFB/TR 14 AVACS and by the DFG/NWO Bilateral Research Program ROCKS.

where probabilistic choices have been replaced by nondeterministic ones. If the PA model is written in MODEST, a backend based on an efficient non-probabilistic model checker such as CADP could be used for a much faster qualitative analysis without first writing an entirely new model in LOTOS. Although quantitative evaluation of the entire STA model spectrum is now within reach [14, 15], the MODEST approach is still as valid as it ever was, since more specialised tools typically provide significant performance gains and often support classes of properties not available in the more general tools.

Paper outline. The first part of this paper focusses on the *single-formalism* aspect with a brief introduction to the MODEST language and the STA formalism (Section 2), including an example from communication protocols to convey a more tangible intuition of the language. After that, we turn to the current state of the *multiple-solution* part, giving an overview of the current tool support for MODEST in Section 3 followed by a survey of previous applications and case studies that used MODEST in Section 4.

2. THE MODEST LANGUAGE

MODEST is an expressive language, both in terms of syntax and semantics. In this section, we first highlight the semantic features, and then look at the language syntax, including a small example.

2.1. Stochastic Timed Automata

Table 1 (adapted from [12, Table IV]) gives an overview of the features and submodels of STA, where *nondet. choices* denotes nondeterministic choices between different edges from the current location, *prob. choices* means that edges lead into (finite-support) probability distributions over target locations and assignments, *nondet. delays* and *stochastic delays* refers to the possibility of specifying delays whose duration is nondeterministic or stochastic according to some probability distribution (where EXP is the exponential distribution), *clock variables* means that more complex time behaviour than nondeterministic or stochastic delays can be specified using clocks, and *compositionality* denotes models for which a natural parallel composition operator exists. The submodels listed are generalised semi-Markov processes (GSMP, [16]), probabilistic timed automata (PTA, [17]), timed automata (TA, [9]), probabilistic automata (PA, [13]) or Markov decision processes (MDP), labelled transition systems, Markov automata (MA, [18]) or interactive Markov chains (IMC, [19]), continuous-time Markov decision processes (CTMDP), as well as continuous- and discrete-time Markov chains (CTMC/DTMC).

MODEST has recently been extended to support the specification and analysis of stochastic hybrid automata (SHA)

	STA	GSMP	PTA	TA	PA/MDP	LTS	MA/IMC	CTMDP	CTMC	DTMC
nondet. choices	✓	–	✓	✓	✓	✓	✓	✓	–	–
prob. choices	✓	✓	✓	–	✓	–	✓	✓	✓	✓
nondet. delays	✓	–	✓	✓	–	–	–	–	–	–
stochastic delays	✓	✓	–	–	–	–	EXP	EXP	EXP	EXP
clock variables	✓	✓	✓	✓	–	–	–	–	–	–
compositionality	✓	–	✓	✓	✓	✓	✓	–	–	–

Table 1. Submodels of stochastic timed automata (STA)

$$
\begin{aligned}
P ::= {} & act \mid \mathsf{stop} \mid \mathsf{abort} \mid \mathsf{break} \mid P_1;\ P_2 \mid \mathsf{when}(e)\ P \mid \\
& \mathsf{urgent}(e)\ P \mid \mathsf{invariant}(e)\ P \mid \mathsf{alt}\ \{\ ::P_1 \ldots ::P_k\ \} \mid \\
& \mathsf{do}\ \{\ ::P_1 \ldots ::P_k\ \} \mid \mathsf{par}\ \{\ ::P_1 \ldots ::P_k\ \} \mid \\
& act\ \mathsf{palt}\ \{\ :e_1\colon asgn_1;\ P_1\ \ldots\ :e_k\colon asgn_k;\ P_k\ \} \mid \\
& ProcName(e_1, \ldots, e_k) \mid \mathsf{throw}(excp) \mid \\
& \mathsf{try}\ \{\ P\ \}\ \mathsf{catch}\ excp_1\ \{\ P_1\ \}\ \ldots\ \mathsf{catch}\ excp_k\ \{\ P_k\ \} \mid \\
& \mathsf{relabel}\ \{\ I\ \}\ \mathsf{by}\ \{\ G\ \}\ \{\ P\ \} \mid \mathsf{extend}\ \{\ H\ \}\ \{\ P\ \}
\end{aligned}
$$

Fig. 1. Syntax of MODEST process behaviours [12]

models [14]. SHA extend STA by adding continuous behaviour governed by differential inclusions. General stochastic hybrid systems are not the focus of this paper; however, a special case are *rewards* or *costs*, where a reward or cost variable increases at a constant rate over time (or by fixed amounts on taking an edge). This allows so-called priced variants of most of the models listed above, for example priced probabilistic timed automata (PPTA, [20]), to be modelled in MODEST as well.

2.2. Syntax and Semantics

MODEST is inspired by process algebras like CCS, CSP or LOTOS. At the core of a MODEST model are processes and *process behaviours*. Processes are names for a process behaviour and a set of local variables. Process behaviours are used to describe the behaviour of a process or a model, such as performing an action, throwing an exception or recursively calling a process. Simple process behaviours are composed into more complex ones using a set of operators such as nondeterministic choice between behaviours (the alt construct) or the sequential composition of two process behaviours (the ; operator). Process behaviours are constructed and composed according to the grammar shown in Figure 1, where act denotes an action name, e ranges over expressions (which are mostly similar to side effect-free expressions from programming languages such as C or ML), $asgn$ ranges over sets of comma-separated assignments (enclosed in $\{= \ldots =\}$ blocks syntactically), and I, G and H are lists of action names.

```
par {
 :: Sender()
 :: relabel { put, get } // sender to receiver
         by { put_data, get_data } Channel()
 :: relabel { put, get } // receiver to sender
         by { put_ack, get_ack } Channel()
 :: Receiver()
}
```

Fig. 2. Overall behaviour of the BRP model in MODEST

The when and urgent constructs are used to add a guard—an enabling condition—or a deadline to an edge in the underlying STA, while invariant adds a location invariant. alt and do allow a *nondeterministic* choice between process behaviours, with do being a looping alt that can be left using break. Similarly, palt represents a *probabilistic* choice over a finite set of alternatives, each of which is given a *weight*. The fraction of an alternative's weight over the sum of all weights of a palt determines its probability. Each alternative also includes an assignment, which in turn can assign randomly sampled values to variables, e.g. $x = \mathrm{Uniform}(0, y)$. The par construct is used to let a set of process behaviours run independently in parallel, possibly *synchronising* on certain actions. In this context, relabeling actions and extending action alphabets can be useful. Finally, throw and try/catch are used to raise and handle exceptions, which is a useful feature imported from programming languages like Java or C# that is not typically supported in a process-algebraic approach.

A formal definition of the semantics of these constructs in terms of structural operational semantics (SOS) rules can be found in [12]. MODEST actually has a two-step semantics: A MODEST model is first mapped to a STA, which is a symbolic model that still contains variables (including clocks) and keeps guards, deadlines, invariants, weights and assignments as (symbolic) expressions. For example, each edge is labelled not only with its action label for synchronisation, but also its guard and deadline (cf. Figure 4). Similar to TA or PTA, the concrete semantics of an STA itself is then defined as an uncountably infinite-state, infinitely-branching timed probabilistic transition system (TPTS).

2.3. Modelling Example

As an example, let us consider the bounded retransmission protocol (BRP, [21]), a communication protocol originally proposed by Philips that adds an upper bound on the number of retransmissions to the well-known alternating bit protocol. We model the scenario of a sender that tries to transmit some file, partitioned into N chunks of data, to a receiver. A natural model of this setting will thus consist of four parts: One component representing the sender, one representing the receiver, and two components representing the communication channel (or: medium) from sender to receiver and vice-versa.

A realistic model of the BRP includes both real-time aspects, namely transmission delays in the channels and the corresponding timeouts on the sender and receiver sides [22], as well as probabilistic choices, namely the loss of messages on the communication channels [23]. The actual delays and loss probabilities to be used will depend on the type of communication channel under study; the loss probability, for example, will be significantly higher for wireless communication compared to, say, a fiber channel link.

```
action put, get;
process Channel() {
  clock c;
  put palt {
    :98: {= c = 0 =};
         // transmission delay in [TMIN, TMAX]
         when(c >= TMIN) invariant(c <= TMAX) get
    : 2: {==} // message lost
  }; Channel()
}
```

Fig. 3. An unreliable communication channel in MODEST

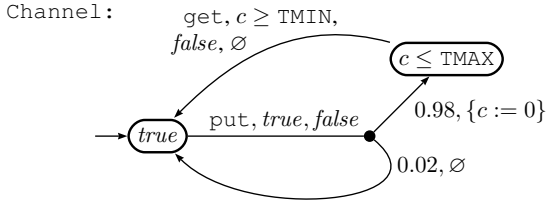

Fig. 4. STA for the unreliable communication channel

The BRP is thus a good example to show how these aspects are combined in MODEST. The four model components can be represented as separate processes that synchronise on certain actions, such as a `put_data` action that is used when the sender puts data on the channel towards the receiver. The process behaviour representing the entire system will then be the parallel composition of one instance of each process (Figure 2), with some renaming of actions in order to use just one process declaration (as a sort of template) for both channel instances. Since the MODEST code for the `Sender` and `Receiver` processes is too large to be shown here[1], let us focus on the communication channel. The MODEST model for the `Channel` process, which synchronises with sender and receiver via actions `put` and `get`, is listed in Figure 3. We see the use of the palt construct to model the probabilistic choice of message loss, and the combination of guards and invariants to specify a nondeterministic transmission delay. The STA semantics for this process is shown in Figure 4.

The properties that we will be interested in for this model fall into four categories:

Invariant properties: We check for absence of premature timeouts ($\mathrm{T_{A1}}$) and that the sender starts a new file only after the receiver has properly reacted to failure ($\mathrm{T_{A2}}$).

[1] The full model of the BRP and models of other case studies are available as part of the MODEST TOOLSET download (see Section 3.1).

Probabilistic reachability: We ask for the worst-case probability that eventually, the sender ($\mathrm{P_A}$) reports a certain unsuccessful transmission but the receiver got the complete file, ($\mathrm{P_B}$) reports a certain successful transmission but the receiver did not get the complete file, ($\mathrm{P_1}$) does not report a successful transmission, and ($\mathrm{P_2}$) reports an uncertainty on the success of the transmission.

Probabilistic time-bounded reachability: We determine the worst-case probability of the sender reporting a successful transmission within 64 time units ($\mathrm{D_{max}}$).

Expected-time reachability: What is the worst-case expected time until the transfer of the first file is finished, successfully or unsuccessfully ($\mathrm{E_{max}}$)?

All properties considered above ask for the *maximum* probability over all resolutions of nondeterminism, which for these properties corresponds to the worst-case scenario; asking for minimum probabilities is, of course, equally possible.

3. TOOL SUPPORT

The guiding principle in tool development for the MODEST language is to reuse existing analysis engines and algorithms where available in order to avoid unnecessary reimplementation work. The first set of MODEST tools, MOTOR [24], provided connections to the MÖBIUS [25] discrete-event simulator and CADP for model checking. Development of a new MODEST TOOLSET, aimed to overcome usability and maintainability shortcomings of MOTOR, started in 2008.

3.1. The Modest Toolset

The MODEST TOOLSET consists of five tools that aid in the modelling process and allow the analysis of MODEST models:

mcpta. The mcpta tool [26, 27] provides model-checking support for models that correspond to the PTA subset of MODEST, i.e. models that do not use any infinite-support probability distributions. By default, it uses a digital clocks semantics [28] to transform PTA into (untimed) PA, which are then analysed using PRISM as a backend. As long as the original PTA are diagonal-free and do not contain clock constraints with strict comparisons (e.g. $c < 3$), this transformation preserves the values of unbounded, time-bounded and cost-bounded probabilistic and expected reachability properties, but in particular not of PTCTL formulae with nested probability operators.

Native support for PTA has recently been added to PRISM [4] and its input language, with analysis support based on both digital clocks as well as a game-based approach [29]. Current versions of mcpta also support the transformation of MODEST models into the new PRISM PTA syntax, as well as automated model-checking using the new game-based analysis engines in PRISM in the background.

mctau. In order to take advantage of UPPAAL's model-checking engine for networks of TA and its graphical modelling and model debugging features, both UPPAAL and MODEST were extended in order to make a transformation between the two languages possible. The result of these efforts is the mctau [30] tool that brings specialised model-checking of TA (which are also covered, though less efficiently, by mcpta) and export to a graphical automata-based formalism (with automatic graph layout) to MODEST. mctau also supports the analysis of PTA models by overapproximating probabilistic choices with nondeterministic ones and modifying the properties of interest accordingly. This is particularly useful for model debugging, since the non-probabilistic analysis is usually significantly faster than the probabilistic one.

modes. The replacement of MOTOR's MÖBIUS-based simulation component in the MODEST TOOLSET is modes [31], a standalone discrete-event simulator for MODEST. The motivation for the development of modes was that simulation cannot be used for nondeterministic models, but most simulation tools—including MOTOR/MÖBIUS—ignore this restriction, using hidden schedulers to resolve the nondeterminism, which may influence the simulation results in unexpected ways [27]. modes in turn refuses nondeterministic models (i.e. models that correspond to the STA, but not the GSMP subset), but allows the schedulers typically used in other tools to be requested explicitly by the user. In order to be able to simulate true STA models without using questionable schedulers, modes also includes a novel partial-order based approach to detect spurious nondeterminism, i.e. nondeterministic choices that to not influence the simulation results, and ignore these [15], thus for the first time allowing a sound treatment of nondeterministic models with a simulation-based tool. A recent addition to modes is support for statistical model checking (SMC, [32, 33, 34]), which is the combination of discrete-event simulation with sequential tests [35].

mime. mime is a graphical user interface that supports editing MODEST models with syntax and error highlighting and that integrates the mcpta, mctau and modes analysis engines in a seamless fashion.

mosta. To give a better understanding of the STA semantics of MODEST code, mosta can be used to transform it into a graphical representation of the underlying STA. mosta uses the Graphviz "dot" tool for automata layout.

Tool availability. The MODEST TOOLSET is freely available for academic users at `www.modestchecker.net`.

	mctau	mcpta	modes
$\mathrm{T_{A1}}$	*true*	*true*	true (all runs satisfied $\mathrm{T_{A1}}$)
$\mathrm{T_{A2}}$	*true*	*true*	true (all runs satisfied $\mathrm{T_{A2}}$)
$\mathrm{P_A}$	0	0	0 (no observations in 10k runs)
$\mathrm{P_B}$	0	0	0 (no observations in 10k runs)
$\mathrm{P_1}$	$[0,1]$	$4.233 \cdot 10^{-4}$	$\mu = 3.0 \cdot 10^{-4}$, $\sigma = 1.7 \cdot 10^{-2}$
$\mathrm{P_2}$	$[0,1]$	$2.645 \cdot 10^{-5}$	0 (no observations in 10k runs)
$\mathrm{D_{max}}$	$[0,1]$	$9.996 \cdot 10^{-1}$	$\mu = 9.9 \cdot 10^{-1}$, $\sigma = 1.7 \cdot 10^{-2}$
$\mathrm{E_{max}}$	n/a	33.473	$\mu = 33.473$, $\sigma = 2.136$

Table 2. Results for the BRP model $(16, 2, 1)$

3.2. Example Analysis

As a first step to analyse our example model of the BRP (here using instance $(16, 2, 1)$, i.e. with a file of 16 chunks, up to 2 retransmissions, and $\texttt{TMAX} = 1$, $\texttt{TMIN} = 0$), we use mctau and its automatic overapproximation of probabilistic choices. Table 2 summarises the results of the different tools; we see that mctau is able to obtain the exact value for the invariant properties and those probabilistic properties that have value 0. mctau cannot handle the expected-time property $\mathrm{E_{max}}$, and cannot safely conclude that the probability for the remaining properties is zero or one.

We then use mcpta to perform a full probabilistic analysis, which takes noticeably longer than the quick check with mctau (still < 1 min for this small example), but yields precise results for all properties considered.

The very same model can also be simulated with modes; however, the results in Table 2 show that this particular model is not very well-suited for simulation because we are interested in rather rare events, some of which were never observed in 10000 simulation runs that also took significantly longer than model-checking. In the table, μ is the mean and σ is the standard deviation of the assumed normal distribution for the results. For property $\mathrm{E_{max}}$, which is not a rare event but an expected value, modes performs very well; in general, the advantage of simulation is that it is not subject to the state-space explosion problem and can thus handle arbitrarily large model instances, as well as more complex properties. We also note that this model has a nondeterministic delay if $\texttt{TMIN} \neq \texttt{TMAX}$; for simulation, we thus set both to 1.

3.3. Outlook

There is a wide range of existing tools that could feasibly be connected with MODEST in a similar way to PRISM (via mcpta) and UPPAAL (via mctau). The main feature that a potential new backend should support is a compositional input formalism that allows a set of automata or modules to run in parallel with a synchronisation mechanism (CSP-style multiway, CCS-style binary or UPPAAL-style broadcast) compatible with MODEST. Connections to non-compositional models are possible (for example, our recent hybrid extensions connect MODEST and PHAVER), but limited to small models due to the size explosion incurred by unrolling parallel composition. That said, an obvious target for the next tool in the MODEST TOOLSET is CADP, to reestablish the bridge that was present in MOTOR and allow model checking of LTS as well as performance evaluation for IMCs, but we are very open to any other suggestion backed by an application need and to collaboration with tool authors.

4. APPLICATIONS

MODEST and its supporting tools have been used in a diverse range of applications, spanning areas such as communication protocols, scheduling, and dependability analysis. We present a survey of the MODEST case studies in this section.

4.1. Communication Protocols

As mentioned in Section 2.3, STA/PTA are a particularly well-suited formalism to model communication protocols, since they usually deal with probabilistic message losses and transmission delays. An early use of MODEST in this area was to analyse two different protocols to monitor nodes in a dynamic network [36] to detect when a node disappears and thus make the network self-configuring. Using the simulation capabilities of the original MOTOR tools, it could be shown that a protocol proposed as an extension of the UPnP standard could lead to unfair and oscillating behaviour that negatively affects power consumption of network devices, while a much simpler protocols avoids these problems.

Abstract models of three communication protocols were also used as case studies for mcpta, since the PTA subset precisely captures the features needed to model most protocols [26]. The protocols studies were the BRP presented in Section 2.3, IEEE 802.11 WLAN, and CSMA/CD as used in Ethernet (IEEE 802.3). In contrast to the BRP, the latter two are inherently probabilistic due to the use of randomised algorithms to resolve contention. Different modelling approaches, including a straightforward pattern to (mechanically) transform a graphical model given as automata into a (text-based) MODEST model, were also explored in that paper.

4.2. Wireless Sensor Networks

In wireless sensor networks (WSN), where small sensor nodes communicate in a self-configuring mesh-style network, the issues typically arising in communication protocol design are compounded by limited computational resources and power constraints, with nodes ideally running for years on small batteries. Protocols to be used in WSN thus have to be both simple and efficient, and their analysis has to take these additional quantitative dimensions into account.

A range of studies has been performed in this area using MODEST, with a prime example of the focus on power

consumption being the analysis of the energy implications of different options of IEEE 802.15.4, which is the basis of e.g. ZigBee. This analysis [37] also took the consequences of drifting clocks into account. It was performed using simulation with MOTOR, as was a later study that focused on latency and, again, energy efficiency, this time comparing the simple and distributed slotted Aloha MAC protocols on different topologies [38]. This later study also used an advanced radio/interference model that is more realistic than the overly simplified unit disk model used in many other models.

Recently, the mcpta and modes tools of the MODEST TOOLSET have been used to evaluate the dependability of a safety-critical system of wireless sensors and actuators with hard real-time requirements [39]: Using both simulation and model checking on a single base model, the safety of a wireless braking system for bikes was proven. The model, in this case, comprised both the link and application layers.

4.3. Dependability

In the realm of dependability analysis, very abstract models of complex systems are typically used to obtain measures for e.g. availability or reliability. It is often desirable to use graphical modelling formalisms specialised for dependability in order to allow domain experts to perform the modelling without needing detailed knowledge of more complex (and more general) formalisms. In this spirit, MODEST itself has been proposed as a backend for two such dependability-focused formalisms: STOCHARTS [40] and ARCADE [41].

As a case study for the use of MODEST in conjunction with STOCHARTS, the GSM-R radio communication used for the next-generation train control system in Europe (ETCS) has been analysed [40]. This system shares many aspects with the communication protocols described above; the model in particular includes probabilistic delays as well as hard real-time requirements.

In evaluating the transformation of ARCADE models to MODEST and the subsequent analysis using the modes simulator [15], two well-known models—a distributed disk architecture, where several hard disks operate in clusters with repairs being performed in a first-come-first-serve manner if disks fail, and a reactor cooling system with a heat exchanger, two pump lines for redundancy and a bypass system—have been used. The partial order-based features of modes were applied in these cases since the models contained spurious nondeterministic choices.

4.4. Renewable Energy Generation

Energy markets and electricity networks are becoming increasingly complex distributed systems through the rapid deployment of microgenerators, in particular in the form of photovoltaic generators on the rooftops of individual homes. Managing electricity networks to balance production and consumption turns out to be a difficult problem as production, in addition to consumption, now becomes a stochastic process (as e.g. the amount of sunlight varies throughout the day, depending on weather and cloud movements).

Current regulations in Germany to control photovoltaic microgenerators in situations of electricity overproduction are simplistic, and may lead to oscillatory behaviour that puts the stability of the electric grid at stake. MODEST and the MODEST TOOLSET have recently been used to study control algorithms for such microgenerators [42, 43]. Inspired by the way computer networks such as the Internet are managed in a decentralised, self-stabilising way, a set of randomised algorithms has been evaluated in terms of grid stability, availability and fairness using the modes simulator. The results indicate that taking ideas from communication protocols to use randomised, distributed control schemes, which also benefit from privacy advantages compared to centralised coordination, may be a promising way to solve the problems inherent in the power grid of the future.

4.5. Industrial Production Scheduling

Finally, in a completely different approach, MODEST and the MOTOR simulation tools have been used to evaluate schedules in a resource-constrained production process [44]. The concrete case study was production of a set of lacquers according to different recipes. Every recipe differs in the kind and order of production steps—and thus in the manufacturing resources—needed, and includes different timing requirements and interdependencies for the individual steps. As a first step, optimal starting times for production jobs were heuristically determined using MODEST models and simulation, using a stochastic model of machine breakdowns and an approximation of resource conflicts through a load model. Based on these starting times, cost-optimised production schedules were generated using the UPPAAL CORA tool [45] from priced timed automata models representing the resource conflicts in a precise way. As a last step, the performance and stability of the generated schedules was evaluated by using stochastic simulation with MODEST and MOTOR again.

5. CONCLUSION

In this paper, we have motivated the need to consider more than pure functional properties when evaluating complex systems. MODEST is a modelling language that aims to unify the construction and analysis process for models that contain quantitative aspects such as probabilistic decisions, real-time behaviour and requirements, and costs such as energy consumption or resource constraints. Based on a single-formalism, multiple-solution approach, the second-generation MODEST TOOLSET provides a growing collection of tools to support modelling and analysis with MODEST. Simulation and model-checking tools for MODEST have

reached a stable state, as evidenced by a number of case studies showing that the language and the tools can be applied to a varied set of problems. Nevertheless, there is still a wide range of work on specialised tools and applications that deserves to be connected to MODEST, either by specifying models in the MODEST language and thus opening up new analysis pathways for interesting and challenging case studies, or by connecting new tools as backends to the MODEST TOOLSET in order to allow them to be used with existing models and to further strengthen the multiple-solution aspect of the MODEST approach.

Acknowledgments. This paper first and foremost builds on work started by Henrik Bohnenkamp, Pedro R. D'Argenio, Holger Hermanns and Joost-Pieter Katoen with the definition of the MODEST language [12] and the creation of the first set of tools [24]. The current MODEST TOOLSET and the theory around its tools was developed in conjunction with Jonathan Bogdoll, Luis María Ferrer Fioriti and Holger Hermanns [15, 26, 31]. The connection between MODEST and UPPAAL would not have been possible without Alexandre David [30]. Ernst Moritz Hahn has also provided valuable input to improve the language semantics during the development of the extension to hybrid systems [14].

6. REFERENCES

[1] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen, "Performance evaluation and model checking join forces," *Commun. ACM*, vol. 53, no. 9, pp. 76–85, 2010.

[2] Hubert Garavel, Frédéric Lang, Radu Mateescu, and Wendelin Serwe, "CADP 2010: A toolbox for the construction and analysis of distributed processes," in *TACAS*. 2011, vol. 6605 of *LNCS*, pp. 372–387, Springer.

[3] Tommaso Bolognesi and Ed Brinksma, "Introduction to the ISO specification language LOTOS," *Computer Networks*, vol. 14, pp. 25–59, 1987.

[4] Marta Z. Kwiatkowska, Gethin Norman, and David Parker, "PRISM 4.0: Verification of probabilistic real-time systems," In *CAV* [46], pp. 585–591.

[5] Ernst Moritz Hahn, Holger Hermanns, Björn Wachter, and Lijun Zhang, "INFAMY: An infinite-state Markov model checker," in *CAV*. 2009, vol. 5643 of *LNCS*, pp. 641–647, Springer.

[6] Ernst Moritz Hahn, Holger Hermanns, Björn Wachter, and Lijun Zhang, "PASS: Abstraction refinement for infinite probabilistic models," in *TACAS*. 2010, vol. 6015 of *LNCS*, pp. 353–357, Springer.

[7] Ernst Moritz Hahn, Holger Hermanns, Björn Wachter, and Lijun Zhang, "PARAM: A model checker for parametric Markov models," in *CAV*. 2010, vol. 6174 of *LNCS*, pp. 660–664, Springer.

[8] Gerd Behrmann, Alexandre David, and Kim G. Larsen, "A tutorial on UPPAAL," in *SFM-RT 2004*. September 2004, number 3185 in LNCS, pp. 200–236, Springer.

[9] Rajeev Alur and David L. Dill, "A theory of timed automata," *Theor. Comput. Sci.*, vol. 126, no. 2, pp. 183–235, 1994.

[10] Nicolas Coste, Hubert Garavel, Holger Hermanns, Frédéric Lang, Radu Mateescu, and Wendelin Serwe, "Ten years of performance evaluation for concurrent systems using CADP," in *ISoLA (2)*. 2010, vol. 6416 of *LNCS*, pp. 128–142, Springer.

[11] Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikucionis, and Zheng Wang, "Time for statistical model checking of real-time systems," In *CAV* [46], pp. 349–355.

[12] Henrik C. Bohnenkamp, Pedro R. D'Argenio, Holger Hermanns, and Joost-Pieter Katoen, "MoDeST: A compositional modeling formalism for hard and softly timed systems," *IEEE Transactions on Software Engineering*, vol. 32, no. 10, pp. 812–830, 2006.

[13] Roberto Segala, *Modeling and Verification of Randomized Distributed Real-Time Systems*, Ph.D. thesis, MIT, Cambridge, MA, USA, 1995.

[14] Ernst Moritz Hahn, Arnd Hartmanns, Holger Hermanns, and Joost-Pieter Katoen, "A compositional modelling and analysis framework for stochastic hybrid systems," *Formal Methods in System Design*, 2012, DOI: 10.1007/s10703-012-0167-z.

[15] Jonathan Bogdoll, Luis María Ferrer Fioriti, Arnd Hartmanns, and Holger Hermanns, "Partial order methods for statistical model checking and simulation," in *FMOODS/FORTE*. 2011, vol. 6722 of *LNCS*, pp. 59–74, Springer.

[16] Peter J. Haas and Gerald S. Shedler, "Regenerative generalized semi-Markov processes," *Communications in Statistics. Stochastic Models*, vol. 3, no. 3, pp. 409–438, 1987.

[17] Marta Z. Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston, "Automatic verification of real-time systems with discrete probability distributions," *Theor. Comput. Sci.*, vol. 282, no. 1, pp. 101–150, 2002.

[18] Christian Eisentraut, Holger Hermanns, and Lijun Zhang, "On probabilistic automata in continuous time," in *LICS*. 2010, pp. 342–351, IEEE Computer Society.

[19] Holger Hermanns, *Interactive Markov Chains: The Quest for Quantified Quality*, vol. 2428 of *LNCS*, Springer, 2002.

[20] Jasper Berendsen, David N. Jansen, and Joost-Pieter Katoen, "Probably on time and within budget: On reachability in priced probabilistic timed automata," in *QEST*. 2006, pp. 311–322, IEEE Computer Society.

[21] Leen Helmink, M. P. A. Sellink, and Frits W. Vaandrager, "Proof-checking a data link protocol," in *TYPES*. 1993, vol. 806 of *LNCS*, pp. 127–165, Springer.

[22] Pedro R. D'Argenio, Joost-Pieter Katoen, Theo C. Ruys, and Jan Tretmans, "The bounded retransmission protocol must be on time!," in *TACAS*. 1997, vol. 1217 of *LNCS*, pp. 416–431, Springer.

[23] Pedro R. D'Argenio, Bertrand Jeannet, Henrik Ejersbo Jensen, and Kim Guldstrand Larsen, "Reachability analysis of probabilistic systems by successive refinements," in *PAPM-PROBMIV*. 2001, vol. 2165 of *LNCS*, pp. 39–56, Springer.

[24] Henrik C. Bohnenkamp, Holger Hermanns, and Joost-Pieter Katoen, "MoTor: The Modest tool environment," in *TACAS*. 2007, vol. 4424 of *LNCS*, pp. 500–504, Springer.

[25] Tod Courtney, Shravan Gaonkar, Ken Keefe, Eric Rozier, and William H. Sanders, "Möbius 2.3: An extensible tool for dependability, security, and performance evaluation of large and complex system models," in *DSN*. 2009, pp. 353–358, IEEE.

[26] Arnd Hartmanns and Holger Hermanns, "A Modest approach to checking probabilistic timed automata," in *QEST*. 2009, pp. 187–196, IEEE Computer Society.

[27] Arnd Hartmanns, "Model-checking and simulation for stochastic timed systems," in *FMCO*. 2010, vol. 6957 of *LNCS*, pp. 372–391, Springer.

[28] Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Jeremy Sproston, "Performance analysis of probabilistic timed automata using digital clocks," *Formal Methods in System Design*, vol. 29, no. 1, pp. 33–78, 2006.

[29] Marta Z. Kwiatkowska, Gethin Norman, and David Parker, "Stochastic games for verification of probabilistic timed automata," in *FORMATS*. 2009, vol. 5813 of *LNCS*, pp. 212–227, Springer.

[30] Jonathan Bogdoll, Alexandre David, Arnd Hartmanns, and Holger Hermanns, "mctau: Bridging the gap between Modest and UPPAAL," in *SPIN*. 2012, vol. 7385 of *LNCS*, pp. 227–233, Springer.

[31] Jonathan Bogdoll, Arnd Hartmanns, and Holger Hermanns, "Simulation and statistical model checking for Modestly nondeterministic models," in *MMB/DFT*. 2012, vol. 7201 of *LNCS*, pp. 249–252, Springer.

[32] Ananda Basu, Saddek Bensalem, Marius Bozga, Benoît Caillaud, Benoît Delahaye, and Axel Legay, "Statistical abstraction and model-checking of large heterogeneous systems," in *FMOODS/FORTE*. 2010, vol. 6117 of *LNCS*, pp. 32–46, Springer.

[33] Håkan L. S. Younes and Reid G. Simmons, "Probabilistic verification of discrete event systems using acceptance sampling," in *CAV*. 2002, vol. 2404 of *LNCS*, pp. 223–235, Springer.

[34] Paolo Zuliani, André Platzer, and Edmund M. Clarke, "Bayesian statistical model checking with application to simulink/stateflow verification," in *HSCC*. 2010, pp. 243–252, ACM.

[35] Abraham Wald, *Sequential analysis*, Wiley, New York, 1959.

[36] Henrik C. Bohnenkamp, Johan Gorter, Jarno Guidi, and Joost-Pieter Katoen, "Are you still there? – a lightweight algorithm to monitor node presence in self-configuring networks," in *DSN*. 2005, pp. 704–709, IEEE Computer Society.

[37] Christian Groß, Holger Hermanns, and Reza Pulungan, "Does clock precision influence ZigBee's energy consumptions?," in *OPODIS*. 2007, vol. 4878 of *LNCS*, pp. 174–188, Springer.

[38] Haidi Yue, Henrik C. Bohnenkamp, Malte Kampschulte, and Joost-Pieter Katoen, "Analysing and improving energy efficiency of distributed slotted aloha," in *NEW2AN*. 2011, vol. 6869 of *LNCS*, pp. 197–208, Springer.

[39] Hernán Baró Graf, Holger Hermanns, Juhi Kulshrestha, Jens Peter, Anjo Vahldiek, and Aravind Vasudevan, "A verified wireless safety critical hard real-time design," in *WOWMOM*. 2011, IEEE.

[40] Holger Hermanns, David N. Jansen, and Yaroslav S. Usenko, "From StoCharts to MoDeST: a comparative reliability analysis of train radio communications," in *WOSP*. 2005, pp. 13–23, ACM.

[41] Hichem Boudali, Pepijn Crouzen, Boudewijn R. Haverkort, Matthias Kuntz, and Mariëlle Stoelinga, "Architectural dependability evaluation with Arcade," in *DSN*. 2008, pp. 512–521, IEEE Computer Society.

[42] Arnd Hartmanns and Holger Hermanns, "Modelling and decentralised runtime control of self-stabilising power micro grids," in *ISoLA*. 2012, LNCS, Springer, to appear.

[43] Arnd Hartmanns, Holger Hermanns, and Pascal Berrang, "A comparative analysis of decentralized power grid stabilization strategies," in *Winter Simulation Conference*, 2012, to appear.

[44] Angelika Mader, Henrik C. Bohnenkamp, Yaroslav S. Usenko, David N. Jansen, Johann Hurink, and Holger Hermanns, "Synthesis and stochastic assessment of cost-optimal schedules," *STTT*, vol. 12, no. 5, pp. 305–318, 2010.

[45] Gerd Behrmann, Kim Guldstrand Larsen, and Jacob Illum Rasmussen, "Optimal scheduling using priced timed automata," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 4, pp. 34–40, 2005.

[46] *Computer Aided Verification - 23rd International Conference, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, vol. 6806 of *LNCS*. Springer, 2011.