QoS Modelling and Analysis with UML-Statecharts: The StoCharts Approach¹

David N. Jansen²

Holger Hermanns^{3,2}

Abstract

The UML is an influential and widespread notation for high-level modelling of information processing systems. UML statechart diagrams are a graphical language to describe system behaviour. They constitute one of the most intensively-used formalisms comprised by the UML. However, statechart diagrams are lacking concepts for describing real-time, performance, dependability and quality of service (QoS) characteristics at a behavioural level.

This note describes a QoS-oriented extension of UML statechart diagrams, called StoCharts. StoCharts enhance the basic statechart formalism with two distinguished features, both simple and easy to understand, yet powerful enough to model a sufficiently rich class of stochastic processes. This is illustrated by a selection of case studies performed using StoCharts. We review the main ingredients of StoCharts and survey tool support and case studies performed with the language, and place StoCharts in the context of other extensions of statechart diagrams.

1 Motivation

The Unified Modelling Language is pervading many challenging engineering areas including real-time and embedded system design. While in principle the UML provides the right ingredients to model discrete-event dynamic systems, it has insufficient support for quality of service (QoS) modelling. Most QoS aspects of current embedded systems are time-related features and properties, and often are of a stochastic nature. A workable modelling and analysis approach to embedded system QoS is based on the observation that networks, interfaces, and even circuits on chips [10, 41] can be understood and modelled as discrete systems exhibiting some form of stochastic behaviour, such as error rates, service time distributions, or message queue length distributions.

Stochastic behaviour has been addressed in both the UML profile for schedulability, performance and time [43], and in

the pending UML profile for modelling QoS and fault tolerance [42]. These profiles suggest annotations of UML providing means to specify performance, dependability and QoS characteristics at various levels.

However, the lack of a formal semantics for the UML and its annotational extensions drastically hampers QoS analysis: It is simply impossible to distill a faithful performance or QoS measure from a stochastic process that is only partially defined. This means that model-based QoS *prediction* is only possible for UML fragments with a rigid formal semantics. For this reason, quite some research has been devoted to formalise QoS analysis based on the UML (see Section 4). Most of this research focuses on the use of UML statechart diagrams.

Statechart diagrams are a graphical language to describe the behaviour of a system. They constitute one of the most intensively used formalisms comprised by the UML. Statechart diagrams are typically used to describe the behaviour of UML objects. Together with Katoen [26], we have recently proposed StoCharts, a QoS-oriented extension of UML statechart diagrams. The design rationales behind this extension have been that the language should be *parsimonious* (in the amount of constructs added), *powerful* (with respect to the QoS models definable), *effective* (with respect to the usefulness in practice and tool support), and *conservative* (extending the original UML statechart diagrams without altering their semantics). These rationales are met by the language StoCharts. StoCharts enhance the basic statechart formalism with two distinguished features, both simple and easy to understand, yet powerful enough to model a sufficiently rich class of stochastic processes. One enhancement allows state transitions to select probabilistically out of different effects, much like the rolling of a die can have one out of six effects, determined probabilistically. The second extension is equally simple: The “after” operator of statechart diagrams is given a stochastic interpretation, allowing the use of arbitrary probability distributions. The resulting statecharts dialect contains standard UML statechart diagrams as a special case.

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

¹This work has been supported by the Dutch and German research councils as part of their bilateral cooperation program, as well as the national programs *SFB/TR-14 AVACS* and *NWO Vernieuwingsimpuls*.

²Formal Methods and Tools, Computer Science Dept., Universiteit Twente, Enschede, Netherlands.

³Dependable Systems and Software, Universität des Saarlandes, Saarbrücken, Germany.

of research in formal specification of stochastic processes, rooted in seminal work on stochastic Petri nets [1], stochastic process algebra [17, 21] and probabilistic automata [37]. In order to associate a stochastic interpretation to collaborative collections of statechart diagrams embedded in arbitrary environments, StoCharts are equipped with a compositional semantics, which uses concepts from Input/Output (I/O) automata [31]. The semantics associated with StoCharts is based on the requirements-level semantics of Eshuis and Wieringa [15], which adapts the Statemate semantics [19] to the UML. The presence of this rigid semantics has enabled us to apply model checking to systems modelled graphically with StoCharts. Model checking is a method to verify automatically whether a system satisfies some requirement. In our context these properties are to be understood as QoS requirements such as: “In at least 95% percent of the cases, the message will arrive within the required interval of 3 to 7 seconds.” As far as we know, this application of model checking to verify QoS requirements is unique in the context of the UML.

This paper reviews the main ingredients of StoCharts and surveys tool support and case studies performed with the language. In Section 2, we introduce the StoCharts language. Section 3 contains some applications we have studied. In Section 4, we compare our work to other stochastic extensions of statecharts. Section 5 concludes the paper.

2 StoCharts

2.1 The UML

The Unified Modelling Language (UML) [36, 33] is a set of languages to describe aspects of a computer system in an object-oriented way. Statecharts [18] have been adopted by the UML to describe the behaviour of objects or other system components. The UML has been successfully introduced into the market of professionally used software specification languages.

The exact *semantics* of the languages in the UML is still under discussion. The UML does not have a fully defined formal semantics; some points are intentionally left open, some other issues are not clarified [35].

The UML includes a mechanism for extension by so-called *profiles*. Examples of extensions are the UML profile for schedulability, performance, and time [43] and the UML profile for QoS and fault tolerance [42].

2.2 UML Statechart Diagrams

Statechart diagrams describe the behaviour of a system similar to an automaton or a state-transition diagram: states are drawn as rounded boxes and transitions as arrows from one state to another, labelled with a triggering event and some action that is executed when the transition is taken. Statechart diagrams extend this simple model in two ways: with *parallelism*,

lelism, one can visually separate the behaviours of (almost) independent components of a system. By means of *hierarchy*, one can refine a behaviour. These extensions make statechart diagrams suitable in situations where simple automata or state-transition diagrams would become cluttered.

Full statechart diagrams contain many more features, like entry and exit actions or history connectors. Most of them can be defined as syntactic sugar and will not play a role in this paper. We only want to mention two: A conditional choice point (graphically visualized by a ©) is a way to choose one out of several possible reactions, based on some condition. A timer event is noted “after(*t*)”, meaning that after a delay of *t* time units, some transition should be taken.

2.3 StoCharts

Traditional statecharts do not include probabilities, although Harel already mentioned “Markov-charts” in his seminal article [18]. We defined StoCharts, an extension of statechart diagrams to model stochastic systems [26]. The extension consists of two points: First, *probabilistic choice* allows transitions to select probabilistically one out of different effects, much like casting a die can have one out of six effects. Second, with *stochastic timing*, one can specify a stochastic delay, for example to describe the duration of some process.

Probabilistic choice. In probabilistic choice, it is uncertain which possibility out of a discrete set the system will take. A probability space over this set describes how likely the single possibilities are. The extension of statecharts with probabilistic choice introduces means to specify such system randomness. Within StoCharts, this probabilistic choice is modelled graphically by a Ⓟ (inspired by the above mentioned © notation) indicating a probabilistic selection of one out of several possible reactions to a triggering event [25]. This probabilistic extension alone already enables a natural and succinct modelling of randomised distributed algorithms, such as the *root contention protocol* in IEEE 1394 [40].

Stochastic timing. In stochastic timing, it is uncertain how long a specific process will take. The timing is described by a probability space over the (nonnegative) real numbers in the form of a probability distribution. For example, a probability distribution can be used to give a prediction on how long a phone call might last. To support this effect, the “after” operator of statechart diagrams is given a stochastic interpretation, allowing the use of arbitrary probability distributions, such as after(EXP[10 min]) for a negative exponential distribution with a mean of 10 min, or after(UNIF[10 h, 15 h]) for a uniform distribution in the interval from 10 to 15 hours. The stochastic timing extension of statecharts thus allows one to quantify the time between events according to a stochastic distribution.

The two extensions of statecharts have been added to provide software engineers (of which many already know basic statecharts) with an easy way to describe probabilistic resp.

Figure 1: Example StoChart

stochastic systems. From a historical perspective this approach is similar in spirit to the generalizations of Petri nets with delays and weights of transitions [1].

Example. Figure 1 shows a StoChart for the workflow of a car damage assessor. His task is to assess on behalf of an insurance company whether a damaged car should be repaired and whether the garage offers an acceptable price for the repair. The StoChart shows that processes may take a shorter or longer time, depending on several circumstances that we do not model. Labels of the form *after*(EXP[*n* min]) indicate that the delay is exponentially distributed with a mean of *n* minutes. Hierarchy is illustrated in state “Assessing”, which is refined into “Physical” and “Phone assessment”. Parallelism is shown in the large state at the bottom, where “Repair” and “Report” are separated by a dotted line. Their refinements are executed in parallel; for example, the system may be in state “Waiting for invoice” and in state “Writing report” at the same time. We suppose the StoCharts to be intuitive enough, such that the assessment process modelled here does not need further explanations.

Semantics. To allow a faithful evaluation of these systems requires a formal semantic model, which is unambiguously defined. The formal semantics of StoCharts is defined as an extension of the statechart semantics of Eshuis and Wieringa [15] and maps on Stochastic Input/Output Automata (IOSA). This is an automata model that combines elements from I/O automata [31], timed automata [2], probabilistic automata [37], and stochastic automata [11]. IOSA are composable, allowing one to combine several StoCharts and to compose them with other languages and models. The details of the semantic mapping from StoCharts to IOSA are out of the scope of this note, we refer to [24, 26] instead. The IOSA model is very expressive, covering a wide range of timed, probabilistic, nondeterministic, and stochastic models. In the spectrum of well-known models one finds Markov

chains (both discrete and continuous time) and decision processes, and nondeterministic timed transition systems. Apart from the input/output distinction (which assures composability), one may regard any IOSA as a generalized semi-Markov decision process [34].

Evaluation. The expressiveness of StoCharts implies that an analysis of general StoChart models is difficult. The presence of both arbitrary probability distributions and nondeterminism (directly inherited from plain UML statechart diagrams) poses challenges. As in the plain UML case, nondeterminism cannot be guaranteed to be absent by construction, but is often absent in practice (as in our case studies). If nondeterminism is absent, the IOSA can be reduced to a generalized semi-Markov process (GSMP) [38]. This model is amenable to efficient discrete-event simulation based evaluation. If, in addition, only negative exponential probability distributions are used, the IOSA can be reduced to a continuous-time Markov chain. Markov chains can be solved with respect to transient and stationary measures using efficient numerical solution techniques [39]. *Stochastic model checking* [3, 4] is an alternative technique, where a precise description of a desired guarantee, usually described using a temporal logic, is verified mechanically, using a *model checker*. This promising technique can be applied to Markov and Markov decision models, and even to GSMPs.

It is important to remark that the IOSA semantics conservatively extends the plain UML statechart semantics without implicitly aiming at a particular solution method. In section 4 we will discuss that many competing dialects of statechart diagrams differ in this respect.

3 Application cases

The attractiveness of the StoChart language can be judged only superficially by looking at its syntax and formal properties. Its effectiveness in practice must be judged on the basis of real-life examples. Since the StoChart formalism has been coined in 2003, we have applied it to case studies of increasing complexity.

Automatic teller machine service quality.

In this case study [24], the QoS characteristics of an automatic teller machine scenario (including bank, customer, and teller machines) were modelled in StoCharts. We focused on a probabilistic requirement of the following form: “Once the card has been entered, the probability that the client obtains the money within time *t* is at least *p*.” This can be considered as a QoS requirement imposed by the bank to ensure customer satisfaction. Figure 2 illustrates the result of our analysis: it shows a typical plot indicating for which values of *t* and *p* the requirement holds. We used model checking to verify the validity of this requirement using the tools ProVer and ETMCC. ProVer [44] estimates probabilities of system requirements up to a user-specified confidence by means of discrete-event sim-

Figure 2: Probability that the withdrawal completes within t seconds

ulation. In models where all distributions were chosen to be exponential, the Markov chain model checker ETMCC [20] was used. The IOSA models associated with the StoChart specifications were generated in a semi-automatic way, and manually fed into the respective model checkers.

This case study has been carried out to provide first experimental evidence that the StoChart approach can be used for modeling and model-based prediction of interesting QoS quantities. [24]

Fruit machine gambling risk.

A fruit machine is a kind of gambling machine that contains several reels showing fruit symbols. When the user starts a game, the reels spin until the user stops them. If the visible parts of the reels show some specific combination of symbols (e. g., three times cherry), the player gets a prize. Gambling machines are subject to legal requirements about the probability distribution over the expected wins and losses, for example, “in 30 games, the player does not lose more than \$ n with probability 0.9.” Traditionally, these requirements have been checked using simulation; however, there is always some (small) chance that the outcome of the simulation gives a wrong estimate of the required probability.

Using probabilistic model checking, we checked some requirements similar to the above for a StoChart model of a fruit machine [23]. With the StoChart language, we were able to describe the parallelism in the fruit machine – three reels spinning independently – nicely. The prize mechanism (which combinations lead to a prize) was easy to model using hierarchy and multi-source transitions. It would have been impossible to get an overview of the system using one of the basic probabilistic languages (without hierarchy or parallelism).

Train radio reliability.

This case study [27] focuses on a safety critical fragment of the European Train Control System (ETCS) standard. This standard aims at ensuring interoperability of European railway systems in the future. Communication among ETCS

components (trains, trackside equipment, etc.) will be based on mobile communication using GSM-R, an adaptation of the GSM protocol to railway applications. Parts of the GSM-R communication is mission-critical, because the train distance management relies on the timely communication of position reports.

We developed a StoChart model of trains and trackside infrastructure components following a design-by-contract approach. The model enabled us to identify bounds on the distance between consecutive trains on a track, under which crucial QoS requirements of ETCS are still satisfied. To arrive at these results, we followed two different paths, which led to consistent results. First, the StoChart collection was manually translated into a simulation model. This model in turn was fed into the tool ProVer. Later, we instead translated the StoChart-model to the language MoDeST [12], supported by the tools MoTor [7] and Möbius [13] for discrete event simulation. Due to the flexibility of the MoTor/Möbius tool chain, we were able to perform a more detailed verification of the QoS requirements met by the GSM-R specification.

Tool support.

From the experience gained through these case studies, it became apparent that the StoCharts approach opens the UML area for model-based prediction of QoS quantities and that StoCharts are a convenient formalism for modelling. However, the examples we studied also showed that the effectiveness heavily relies on seamless tool support, ranging from graphical specification of StoCharts to presentation of analysis results in two- or three-dimensional plots. Therefore, tool support for StoChart needs to improve.

While drawing tools for UML statechart diagrams and StoCharts are at hand (e. g. TCM [14]), analysis tools which can digest StoChart designs directly are missing. Therefore we have decided to invest in a tool environment for StoCharts. Instead of starting a new tool development, we decided to bridge to the ongoing activities in the context of the MoTor tool [7]. MoTor is linked to Möbius [13] for discrete-event simulation-based analysis, and it uses the modeling and specification language MoDeST as an input language. MoDeST is a formal language to describe stochastic timed systems [12], equipped with a rigid formal semantics. The functional core of MoDeST can be considered as a simple process algebra enriched with some convenient language constructs, and a C-like notation. This core language is enriched with several modelling concepts tailored to model timed and/or stochastic systems. MoDeST has been successfully used in a number of nontrivial case studies, see for example [8].

4 Related Approaches to Stochastics in Statecharts

Statecharts were introduced by Harel [18], who mentioned the possibility of a probabilistic extension, but did not detail this. Several other groups proposed similar extensions

of statecharts. One research strand maps variations of statecharts to kinds of stochastic Petri nets: Bernardi, Donatelli, and Merseguer [6] present a translation from a number of statecharts, combined with a sequence diagram, to a generalised stochastic Petri net. King and Pooley [28] describe a system using simplified UML collaboration diagrams and statecharts, similar to our use of the UML. Huszerl et al. [22] define a guarded statecharts dialect where exponentially distributed times govern the triggering of the guarded statechart by the (otherwise unspecified) environment. Lindemann et al. [30] introduce timed events which are similar to our after pseudo-events. They analyse a fixed collection of UML statecharts by defining a semantics in terms of a GSMP. All these approaches have in common that by mapping on stochastic Petri nets, the nondeterminism often inherent to the system specification is replaced by probabilism, while it is kept explicit in the IOSA model. Furthermore, most extensions restrict to a simplified language and/or use a more involved syntax than StoCharts.

Mitton and Holton [32] and Cavenet et al. [9] present translations from statecharts to the stochastic process algebra PEPA [21]. They restrict to exponential distributions and deviate from the standard semantics of UML statechart diagrams. Gnesi, Latella and Massink [16] describe an extension of a hierarchical automaton model for statecharts [29] which is similar in spirit to StoCharts, but exposes the engineer to some low-level semantic ingredients which we hide in the after-construct.

In summary, StoCharts are particular because they are *parsimonious* in the amount of constructs added, *powerful* with respect to the QoS models definable, and *effective* with respect to the usefulness in practice and tool support. None of the other approaches possesses these properties to the extent StoCharts do.

Balsamo and Simeoni [5] provide a more general overview of several performance analysis methods based on UML models, not restricted to statecharts. In this more general context, the UML profile for schedulability, performance, and time [43] is intended to enable quantitative predictions of time-, schedulability- and performance-related aspects of a real-time system. Partly overlapping with these goals is the UML profile for quality of service and fault tolerance [42] (currently being finalized by the OMG). These profiles introduce annotational extensions to indicate performance and other QoS characteristics of systems modelled in UML. Annotations can for instance indicate how long an activity lasts or should last. No effort is made to render precisely the meaning of these annotations. To turn these annotations into a modelling feature which can be subject to some well-defined analysis implicitly requires to make precise the semantics of these annotations. As a starting point one may try to map them onto StoCharts or IOSA.

5 Conclusion

In this note we have reviewed the StoCharts approach to QoS-oriented modelling with UML. StoCharts allow one to use arbitrary delay distributions and probabilistic alternatives inside an arbitrary UML statechart diagram. A rigorous formal semantics defines a mapping on IOSA, a composable low-level model amenable to different types of model analysis, e. g., model checking and discrete-event simulation. Our case studies have emphasized the practicality of the approach, but have also revealed that seamless tool support is very important. We are working on this seamless tool chain, from visual specification to automatic model analysis and result presentation.

References

- [1] M. Ajmone Marsan, G. Conte, and G. Balbo. A class of generalised stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Trans. on Comp. Sys.*, 2(2):93–122, 1984.
- [2] R. Alur and D. Dill. A theory of timed automata. *Theor. Comp. Sci.*, 126:183–235, 1994.
- [3] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Trans. on Softw. Eng.*, 29(6):524–541, 2003.
- [4] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. Model checking meets performance evaluation. In this volume.
- [5] S. Balsamo and M. Simeoni. On transforming UML models into performance models. In *Workshop on Transformations in UML 2001*, 6 p., <http://ase.arc.nasa.gov/wtum101/>.
- [6] S. Bernardi, S. Donatelli, and J. Merseguer. From UML sequence diagrams and statecharts to analysable Petri net models. In *Proc. 3rd Intl. Workshop on Software and Performance*, 35–45, New York, 2002. ACM Press.
- [7] H. Bohnenkamp, H. Hermanns, J.-P. Katoen, and R. Klaren. The MoDeST modeling tool and its implementation. *LNCS 2794*:116–133.
- [8] H. Bohnenkamp, H. Hermanns, J. Klaren, A. Mader, and Y. S. Usenko. Synthesis and stochastic assessment of schedules for lacquer production. In *QEST 2004*, 28–37. IEEE CS Press, 2004.
- [9] C. Cavenet, S. Gilmore, J. Hillston, and P. Stevens. Performance modelling with UML and stochastic process algebra. In *UKPEW 2002*, 12 p., 2002.
- [10] C. Constantinescu. Impact of deep submicron technology on dependability of VLSI circuits. In *DSN 2002*, 205–209. IEEE CS Press, 2002.
- [11] P. R. D’Argenio. *Algebras and automata for timed and stochastic systems*. PhD thesis, Universiteit Twente, Enschede, November 1999.

- [12] P. R. D'Argenio, H. Hermanns, J.-P. Katoen, and R. Klaren. MoDeST : a modelling and description language for stochastic timed systems. *LNCS* 2165:87–104.
- [13] D. D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. G. Webster. The Möbius framework and its implementation. *IEEE Trans. on Softw. Eng.*, 28:956–969, 2002.
- [14] F. Dehne, H. van de Zandschulp, and R. Wieringa. Toolkit for conceptual modeling (TCM). <http://www.cs.utwente.nl/~tcm/>.
- [15] R. Eshuis and R. Wieringa. Requirements-level semantics for UML statecharts. In *FMOODS IV*, 121–140, Boston, 2000. Kluwer Academic Publishers.
- [16] S. Gnesi, D. Latella, and M. Massink. A stochastic extension of a behavioural subset of UML statechart diagrams. In *HASE 2000*, 55–64. IEEE CS Press, 2000.
- [17] N. Götz, U. Herzog, and M. Rettelsbach. Multiprocessor and distributed system design : the integration of functional specification and performance analysis using stochastic process algebras. *LNCS* 729:121–146.
- [18] D. Harel. Statecharts : a visual formalism for complex systems. *Sci. of Comp. Prog.*, 8(3):231–274, 1987.
- [19] D. Harel and A. Naamad. The STATEMATE semantics of statecharts. *ACM Trans. on Softw. Eng. and Meth.*, 5(4):293–333, 1996.
- [20] H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. A tool for model-checking Markov chains. *Intl. Journal on Softw. Tools for Techn. Transfer*, 4(2):153–172, 2003.
- [21] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, Cambridge, 1996.
- [22] G. Huszerl, I. Majzik, A. Pataricza, K. Kosmidis, and M. Dal Cin. Quantitative analysis of UML statechart models of dependable systems. *The Computer Journal*, 45(3):260–277, 2002.
- [23] D. N. Jansen. Probabilistic UML statecharts for specification and verification : a case study. In *CSDUML 2002*, Technical Report TUM-I0208, 121–131, TU Munich, 2002.
- [24] D. N. Jansen. *Extensions of statecharts with probability, time, and stochastic timing*. PhD thesis, Universiteit Twente, Inmarks, Bern, October 2003.
- [25] D. N. Jansen, H. Hermanns, and J.-P. Katoen. A probabilistic extension of UML statecharts : specification and verification. *LNCS* 2469:355–374.
- [26] D. N. Jansen, H. Hermanns, and J.-P. Katoen. A QoS-oriented extension of UML statecharts. *LNCS* 2863:76–91.
- [27] D. N. Jansen and H. Hermanns. Dependability checking of train radio signalling using UML statecharts. In *QEST 2004*, 250–259. IEEE CS Press, 2004.
- [28] P. King and R. Pooley. Derivation of Petri net performance models from UML specifications of communications software. *LNCS* 1786:262–276.
- [29] D. Latella, I. Majzik, and M. Massink. Automatic verification of a behavioural subset of UML statechart diagrams using the SPIN model-checker. *Formal Aspects of Computing*, 11(6):637–664, 1999.
- [30] C. Lindemann, A. Thümmler, A. Klemm, M. Lohmann, and O. P. Waldhorst. Performance analysis of time-enhanced UML diagrams based on stochastic processes. In *Proc. 3rd Intl. Workshop on Software and Performance*, 25–34, New York, 2002. ACM Press.
- [31] N. A. Lynch and M. R. Tuttle. An introduction to input/output automata. *CWI Quarterly*, pages 219–246, 1989.
- [32] P. Mitton and R. Holton. PEPA performability modelling using UML statecharts. In *UKPEW 2000*, 15 p., 2000.
- [33] *OMG unified modeling language specification, version 1.5*. Object Management Group, Needham, MA, 2003.
- [34] M. L. Puterman. *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. Wiley, New York, 1994.
- [35] G. Reggio and R. J. Wieringa. Thirty one problems in the semantics of UML 1.3 dynamics. In *OOPSLA 99 Workshop on Rigorous Modeling and Analysis with the UML*, 1999.
- [36] J. Rumbaugh, I. Jacobson, and G. Booch. *The unified modeling language reference manual*. Addison Wesley, Reading (Mass.), 1998.
- [37] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.
- [38] G. S. Shedler. *Regenerative stochastic simulation*. Academic Press, Boston, MA, 1993.
- [39] W. J. Stewart. *Introduction to the numerical solution of Markov chains*. Princeton University Press, Princeton, NJ, 1994.
- [40] M. Stoelinga and F. Vaandrager. Root contention in IEEE 1394. *LNCS* 1601:53–74.
- [41] D. Tennenhouse. Proactive computing. *Comm. of the ACM*, 43(5):43–50, 2000.
- [42] *UML profile for modeling quality of service and fault tolerance characteristics and mechanisms*. Object Management Group, Inc., Needham, MA, September 2004. <http://www.omg.org/cgi-bin/doc?ptc/2004-09-01>.
- [43] *UML Profile for Schedulability, Performance, and Time Specification : OMG Adopted Specification*. Object Management Group, Inc., Needham, MA, November 2002. <http://www.omg.org/cgi-bin/doc?ptc/2002-03-02>.
- [44] H. L. S. Younes and R. G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. *LNCS* 2404: 223–235.