Project no.:	**ICT-FP7-STREP-214755**
Project full title:	**Quantitative System Properties in Model-Driven Design**
Project Acronym:	**QUASIMODO**

Deliverable no.: D1.2

Title of Deliverable: Design Notations

Contractual Date of Delivery to the CEC:	**Month 6**
Actual Date of Delivery to the CEC:	**Month 24 (February 1, 2010)**
Organisation name of lead contractor for this deliverable:	**Saarland University**
Author(s):	**Pepijn Crouzen,** **Holger Hermanns,** **Jacob Illum** **Joost-Pieter Katoen,** **Arne Skou**
Participants(s):	**P01 AAU, P02 ESI, P04 RWTH, P05 SU**
Work package contributing to the deliverable:	**WP 1**
Nature:	**R**
Version:	**2.2**
Total number of pages:	**9**
Start date of project:	January 1, 2008 **Duration:** 36 month

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)

Dissemination Level

PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract:

In order to facilitate an integration of the techniques developed within Quasimodo into the embedded software engineering life-cycle, we are investigating how to embed them into contemporary design notations. On the one hand side, we are focussing on Architectural Description Languages, and on the other hand we are targetting Statecharts, a design notation based on hierarchical state machines which is widely used for embedded software design in, for instance, the automotive and avionics industry.

Contents

1 Introduction

The UML is pervading many challenging engineering areas including real-time and embedded system design. Embedded systems designers are usually facing various challenges when striving for systems with *quantifiable quality of service* (quantifiable QoS). Most QoS aspects of current embedded systems are time-related features and properties. They may be hard or soft real-time constraints, and are often a of stochastic nature. To incorporate these constraints in the embedded systems design process is a challenging issue.

A workable modeling and analysis approach to quantifying embedded system QoS is based on the observation that networks, interfaces, and even circuits on chips can be understood and modeled as discrete systems exhibiting some form of timed and stochastic behavior, such as reaction times, error rates, response time distributions, communication channel failures or message queue lengths.

Mathematically speaking, the QoS characteristics of a given embedded system induce families of stochastic decision processes, e. g. (Semi) Markov chains, Markov decision processes, or probabilistic timed automata. Workpackage 2-4 of Quasimodo contribute to advances in theory and analysis of such models. However, these mathematical objects are too fine-grained to be directly used as specification means by an embedded systems designer. The modeling of realistic embedded systems directly in terms of these flat automata-based models is unmanageable; even for simple systems, the size and complexity gets out of hand. Therefore, high-level modeling techniques with accompanying tools are needed for stochastic processes. To enable the use of such techniques by system engineers, the modeling notation must be simple, easy-to-use, and closely fit with the techniques already used.

In this deliverable, we report on progresses within Task 1.3, on design notations for reactive systems, and their extensions towards quantitative information. The *Description of Work* is centered around extensions of UML-Statecharts, a modeling technique that is heavily used in embedded system design such as the automotive and aerospace industry. The tasks have originally been tailored to the needs of the industrial tool provider Inchron GmbH, an original partner in the consortium who decided to drop out at project start time. Inchron is a SME tool provider developing simulation tools for execution time- and schedulability- analysis of embedded systems.

While we continued to investigate Statechart-based notations, this drop out implied that the concrete goals and especially the notation to be used were no longer definite. Therefore, we instead explored different industrial Statechart dialects with respect to their extensability toward quantitative information, from different perspectives. These activities were complemented by investigations in the context of Architectural Description Languages (AADLs), due to encouraging input from the European Space Agency (ESA). The Quasimodo project made considerable progress in this context, as we will report below.

2 Extension of Statecharts

2.1 StoCharts: Formality and Toolability

Participants

- David N. Jansen, ESI: Radboud University, Nijmegen, the Netherlands
- Jonathan Bogdoll and Holger Hermanns, Saarland University, Germany
- Joost-Pieter Katoen, RWTH Aachen, Germany

Contributions In principle, the UML provides the right ingredients to model discrete event dynamic systems. It however lacks support for stochastic process modeling. This issue has been addressed in the UML profiles for schedulability, performance and time [13], and for modeling QoS and fault tolerance [12]. It is also touched upon in the more recent MARTE profile on real-time and Embedded system [8]. These profiles suggest annotational extensions of UML providing means to specify performance, dependability and QoS characteristics at various levels. However, the imprecise semantics of the UML and of its annotational extensions drastically hampers *trustworthy* QoS analysis: It is simply impossible to distill a faithful performance or QoS quantity from a stochastic (decision) process that is only partially defined. This means that model-based QoS *prediction* is only possible for UML fragments with a rigid formal semantics. Only in this way, the mathematical stochastic object specified at a high level of abstraction can be uniquely and precisely determined, and thus analysed.

In this work we focus on Statecharts, taking up earlier work in this direction [11] called StoCharts. We work with an extension of Statecharts that is both simple and effective. We enrich the modelling power of Statecharts by, in addition to non-deterministic choice, supporting a probabilistic choice operator is introduced, and the after-construct is generalized such that delays are no longer restricted to be deterministic, but they maybe random. We further allow cost decoration in states. These three simple ingredients allow for the modeling of a rich class of random, timed, and cost phenomena.

We have made strong efforts to design the language in such a way that is itself a useful vehicle in QoS modeling and prediction, and lends itself directly to an implementation that can harvest the tool support provided by the Quasimodo consortioum. This requires more than an easy-to-grasp extension of UML with an intuitive interpretation. In order to support model-based QoS prediction, a rigid formal semantics, because only this enables trustworthy model-based predictions about the actual system. Concretely, we contributed a semantic mapping which conservatively extends the standard semantics of Statecharts. This semantics is compositional, and exploits lessons learned in a decade of research in formal specification of stochastic The semantics is directly implementable, and enables tool-supported analysis and evaluation. This has been exemplified with a very prototypical tool chain that maps on StoCharts the MoDest language for which tool support is being developed in Quasimodo. Experimental evidence shows that this approach can indeed be used for model-based prediction of QoS quantities.

This work is reported in [10].

Perspective The availability of a rigid compositional semantics for this extension of UML-Statecharts, plus initial tool support is a great step ahead, we feel. There is however an interesting feature to be investigated further: The model resulting from a StoChart is a MoDest specification, and often exhibits all of the following three ingredients: continuous stochastic distributions, fixed delays, and nondeterminism. This means that the model is outside the class that can currently be analysed, since the Quasimodo tool family is restricted to models that either do not exhibit nondeterminism (in this case discrete event simulation is used), or at most exhibit discrete probabilism (leading to probabilistic timed automata, whence Quasimodo tools, especially mcpta [9] are resorted to). This issue is further investigated as part of Workpackage 3.

2.2 UPPAAL modelling and Concurrent Statecharts

Participants

- Kim Larsen, Alexandre David, Jacob Illum, and Arne Skou, Aalborg University, Denmark

Contributions Timed Automata constitute one of the modelling formalisms that is used in the Quasimodo project for analyzing quantitative system aspects, and in order to gain experience on tool chain integration when applying UML Statecharts as the design notation for modeling and specification, AAU has conducted an experiment on model transformation from Statecharts to Timed Automata. The involved tools are Rational Systems Developer (Statecharts) and UPPAAL (Time Automata), and the intended use of the transformation is automatic test generation from Statechart models of the MMI part of an embedded device.

The translation is done by first exporting the UML Statecharts into the standard XMI format, and then translating into the UPPAAL XML format for Timed Automata. The Timed Automata is then analysed by using the UPPAAL analyzing engine, and finally the derived Timed Automata traces are translated into the actual test scripting language (JavaScript) by interpreting the UML signals, time events, and state names.

The following restrictions and extensions of the Statechart models are considerend:

- Concurrent Statecharts are not allowed in order to simplify the learning curve for the modelers.
- 'Supersteps' are ignored in the translation in order to simplify the model transformation.
- UML signals are interpreted as user inputs to the MMI system when executing test cases.
- UML time events are interpreted as timeouts when executing test cases.
- UML state names are interpreted as observations from the embedded device when executing test cases.
- UML comments are translated into UPPAAL declarations (types, variables, clocks, function declarations). This means that the syntax check is made by UPPAAL.

- Guards and assignments on UML transitions are translated into guards and assignments in UPPAAL transitions (and syntax checked by UPPAAL).

Perspective The experiment has shown that for Timed Automata there is a straightforward way to translate from Statecharts, and this is translation is indeed being applied for analyzing and testing the Terma industrial case.

2.3 Markov decision processes and STATEMATE

Participants

- Holger Hermanns, Reza Pulungan, Saarland University, Germany

Contents In cooperation with the German special research initiative AVACS we continued our efforts to extend the STATEMATE tool and notation. We finished a (plug-in) extension tailored to the evaluation of quantitative dependability properties at design time. We map on continuous-time Markov decision processes, for which we enable timed reachability analysis.

The extension is compositional in the way the model is augmented with stochastic information. This means that the modelling consists of two parts, where one is a plain – unaltered – Statechart, and the other is a collection of small Markov automata, each carrying the relevant information how a particular sequence of events is interspersed in time. Their joint semantics constrains the behaviour of the original Statechart model such that the timing adheres to the given additional information.

The compositionality is exploited in the construction of the underlying model. The entire tool flow has been implemented and applied to a nontrivial example of a high-speed train signalling system [1].

Perspective We consider this work as a possible blueprint for an extension of an almost arbitrary given Statechart dialect, with timed, cost, stochastic or hybrid features. This is possible since the approach leaves the chosen original Statechart semantics untouched. Instead, it interfaces on the level of the underlying transition system semantics and allows for a notationally crisp and lightweight specification of quantitative information.

3 AADL-based notations

Participants

- Hichem Boudali, Boudewijn R. Haverkort, Matthias Kuntz and Marielle Stoelinga, ESI: Universiteit Twente, the Netherlands
- Pepijn Crouzen, Saarland University, Germany
- Joost-Pieter Katoen and Viet Yen Nguyen, RWTH Aachen, Germany

Contribution Hardware/software (HW/SW) co-design of safety-critical embedded systems such as on-board systems that appear in the aerospace domain is a very complex and highly challenging task. Component-based design is an important paradigm that is helpful to master this design complexity while, in addition, allowing for reusability. The key principle in component-based design is a clear distinction between component behavior (implementation) and the possible interactions between the individual components (interfacing). Components may be structured in a hierarchical manner akin to an AND-composition in the visual modeling formalism Statecharts. The internal structure of a component implementation is specified by its decomposition into subcomponents, together with their HW/SW bindings and their interaction via connections over ports. Component behavior is typically described by a textual representation of mode-transition diagrams, a kind of finite-state automata.

As safety-critical systems are subject to hardware and software faults, the adequate modeling of faults, their likelihood of occurrence, and the way in which a system can recover from faults, are essential to a model-based approach for safety-critical systems. Although several formal approaches to component-based design have been recently reported in the literature, error handling and modeling has received scant attention, if at all. Another shortcoming of many proposals — notable exception is the recent work of [7] — is the lack of connection to a notation that is used and understood by design engineers. Within the context of Quasimodo, and an accompanying project supported by the European Space Agency (ESA), we attempt to overcome these shortcomings by enriching a practical component-based modeling approach with appropriate means for modeling probabilistic fault behavior.

To warrant acceptance by design engineers in, e.g., aerospace industry and the automotive engineers, our approach is based on the Architecture Analysis and Design Language (AADL), a design formalism that is standardized by the Society of Automotive Engineers. The major distinguishing aspects of AADL are the possibility to describe nominal hard- and software operations, hybrid (and timing) aspects, as well as dynamic reconfiguration of components and port connections between components.

In order to model probabilistic faults, their propagation and recovery, and degraded modes of operation, we adopt the recent AADL Error Model Annex. The paper [4] provides a gentle introduction to the resulting language by means of a small example, and presents in detail a formal semantics of a significant subset of extended AADL that provides the interpretation of these model specifications in a precise and unambiguous manner. As a semantical model for the nominal system behavior we use networks of event-data automata. Such automata are in

fact symbolic means to model (besides the usual automata ingredients) discrete data variables and continuous evolution such as the advance of time and variables whose temporal behavior is described by differential equations.

Error behavior is defined by probabilistic finite-state machines, where error delays are determined by continuous random variables, in particular, those that are governed by negative exponential distributions. This strongly resembles the well-studied model of continuous-time Markov chains (CTMCs), with the exception that nondeterminism is also allowed in our setting. The integration of nominal behavior and error models basically boils down to a product construction of an event-data automaton and a finite interactive Markov chain (IMC) (a variant of CTMC with nondeterminism).

In our work on Arcade [2, 3], we use so called deep compositionality, not only is the syntax compositional (i.e. models can be built by combining building blocks), but the semantics is as well compositional. Each syntactical element has its own input-output IMC behavior. The behavior of an entire model then emerges as the composition of the component behaviors. This allows the use of compositional aggregation to mitigate the state space explosion problem. Moreover, deep compositionality allows Arcade to be easily extended. New syntactical elements can be added by defining their behavior as an input/output-IMC model.

Perspective The rigid mathematical foundation as provided in [4, 2] yields a solid basis for developing software tools to support the modeling and formal analysis of AADL specifications, also enabling advanced features such a compositional aggregation [3]. The tool currently under development includes safety and dependability analysis (FMEA and fault tree analysis), model checking, performance evaluation using probabilistic model checking, and fault diagnosability (FDIR). The unique character of this toolset [5] is that it provides all these formal analyses in a single uniform framework. Currently, the approach is applied by a major industrial partner to satelite software, and the language extensions to AADL, as well as its semantics are currently under investigation by the AADL standardization bodies. To the best of our knowledge, this is the first AADL toolset that supports this large variety of analyses.

Bibliography

[1] Eckard Böde, Marc Herbstritt, Holger Hermanns, Sven Johr, Thomas Peikenkamp, Reza Pulungan, Ralf Wimmer, Bernd Becker. Compositional Reliability Evaluation for STATEMATE. *IEEE Transactions on Software Engineering*, Vol. 35, No. 2, 274-292, 2009.

[2] Hichem Boudali, Pepijn Crouzen, Boudewijn R. Haverkort, Matthias Kuntz, Marielle Stoelinga. Architectural dependability evaluation with Arcade. In *DSN 2008*, 512-521, IEEE CS press, 2008.

[3] Hichem Boudali, Pepijn Crouzen, Boudewijn R. Haverkort, Matthias Kuntz, Marielle Stoelinga. Arcade - A Formal, Extensible, Model-Based Dependability Evaluation Framework. In *ICECCS 2008*, 243-248. IEEE CS press, 2008.

[4] Marco Bozzano, Alessandro Cimatti, Marco Roveri, Joost-Pieter Katoen, Viet Yen Nguyen, and Thomas Noll. Codesign of Dependable Systems: A Component-Based Modeling Language. In *Proc. 7th ACM-IEEE Int. Conf. on Formal Methods and Models for Codesign (MEMOCODE 2009)*. ACM Press, 2009.

[5] Marco Bozzano, Alessandro Cimatti, Joost-Pieter Katoen, Viet Yen Nguyen, Thomas Noll, and Marco Roveri. Verification and performance evaluation of aadl models (tool demonstration). In *Proc. 7th Joint Meeting of European Software Engineering Conference and ACM SIGSOFT Symp. on the Foundations of Software Engineering (ESEC/FSE 2009)*, pages 285–286. ACM Press, 2009.

[6] Marco Bozzano, Alessandro Cimatti, Joost-Pieter Katoen, Viet Yen Nguyen, Thomas Noll, and Marco Roveri. The COMPASS Approach: Correctness, Modelling and Performability of Aerospace Systems. In *28th Int. Conf. on Computer Safety, Reliability and Security (SAFECOMP 2009)*, pages 173-186. Volume 5775 of LNCS. Springer, 2009.

[7] M.Y. Chkouri, A. Robert, M. Bozga, and J. Sifakis. Translating AADL into BIP – application to the verification of real-time systems. In *Models in Software Engineering, Workshops and Symposia at MODELS 2008*, pages 39-53. Volume 5421 of LNCS. Springer, 2008.

[8] S. Gurard, D. Petriu and J. Medina. MARTE: A New Standard for Modeling and Analysis of Real-Time and Embedded Systems In 19th Euromicro Conference on Real-Time Systems (ECRTS 07), Pisa, Italy, July 2007.

[9] Arnd Hartmanns, Holger Hermanns. A Modest Approach to Checking Probabilistic Timed Automata. In *Sixth International Conference on the Quantitative Evaluation of Systems (QEST)*, IEEE Computer Society Press, 2009.

[10] Holger Hermanns, David N. Jansen, and Joost-Pieter Katoen. *StoCharts: QoS Modeling and Analysis with UML Statecharts*, in preparation, 2010.

[11] David N. Jansen, Holger Hermanns, and Joost-Pieter Katoen. A QoS-oriented extension of UML statecharts. In Perdita Stevens, Jon Whittle, and Grady Booch, editors, *«UML» 2003 : the unified modeling language*, volume 2863 of *LNCS*, pages 76–91, Berlin, 2003. Springer.

[12] *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`.

[13] *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`.