



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 – preliminary version

Contractual Date of Delivery to the CEC:	Month 6
Actual Date of Delivery to the CEC:	Month 12 (February 1, 2009)
Organisation name of lead contractor for this deliverable:	Saarland University
Author(s):	Pepijn Crouzen, Holger Hermanns, 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:	8
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 targeting 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.

This is a preliminary version of Deliverable 1.2. The final version is being delayed for organisational reasons.

Contents

Abbreviations	2
1 Introduction	3
2 AADL-based notations	3
2.1 AADL+Error specifications	3
2.2 Arcade design notation	4
3 Extension of Statecharts	5
3.1 UPPAAL modelling and Concurrent Statecharts	5
3.2 Markov decision processes and STATEMATE	6
4 Library of component models	6
5 Modelling tool improvements	7
Bibliography	8

Abbreviations

AAU: Aalborg University, DK

ESI: Embedded Systems Institute, NL

ESI/UT: Universiteit Twente, NL

RWTH: RWTH Aachen University, D

SU: Saarland University, D

1 Introduction

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* lists the following items

1. Extension of UML-Statecharts
2. Library of component models
3. Modelling tool improvements

These task items have originally been tailored to the needs of the industrial tool provider Inchron GmbH, who decided to drop out of the consortium at start time (see periodic progress report). 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, the task on libraries of component models as well as improvements to modelling tools are both facing delays.

As a consequence of the required change in task contents, the delivery of the final version of D1.2 is delayed to month 24.

In the first year of Quasimodo, we explored two different industrial Statechart dialects with respect to their extensibility toward quantitative information, from different perspectives. These activities were complemented by additional investigations in the context of Architectural Description Languages (AADLs), as reported below.

2 AADL-based notations

2.1 AADL+Error specifications

Participants

- Joost-Pieter Katoen and Viet Yen Nguyen (RWTH)

Contribution We investigated the design notation AADL (The Architecture Analysis and Design Language) as released in November 2004 by the Society of Automotive Engineers (SAE), and in particular its error annex which allows for the modeling of error behavior. The resulting design notation is tailored to the specific characteristics of critical on-board embedded systems for the space domain. The design notation offers engineers with convenient ways to specify a.o. nominal hardware, as well as software operations, (probabilistic) faults and their propagation, error recovery and degraded modes of operation. A kernel of the design notation is equipped

with a formal semantics that provides the interpretation of AADL+error specifications in a precise and unambiguous manner. Systems are considered as a hierarchy of (hardware and software) components where components are defined by their type (interface) and implementation. Components communicate via ports allowing for message and continuous communication. 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 described by a textual description of mode-transition diagrams. System reconfiguration is supported by mode-dependent presence of subcomponents and their connections. Error behaviour is described by probabilistic finite state machines, where error delays may be governed by continuous random variables. The combination of nominal and error behaviour has been investigated and formalized in the formal (operational) semantics.

Correctness properties, safety guarantees, and performance and dependability requirements will be specified using requirement specification patterns which act as parameterized "templates" to the engineers and thus offer a comprehensible and easy-to-use framework for requirement specification. The properties will be checked on the AADL specification using formal analysis techniques such as model checking and probabilistic variants thereof.

More details can be found in [4]. A conference submission with further details is being prepared.

Perspective The precise meaning of the design notation is a prime feature of this work, yielding a trustworthy modeling and analysis framework for system and software engineers.

2.2 Arcade design notation

Participants

- Hichem Boudali, Boudewijn R. Haverkort, Matthias Kuntz and Marielle Stoelinga (ESI/UT)
- Pepijn Crouzen (SU)

Contribution The Arcade (Architectural Dependability Evaluation) approach combines the strengths of several previous approaches to provide a powerful dependability analysis tool. The key feature of Arcade is its rigorous semantics in terms of input/output interactive Markov chains (I/O-IMCs). Combined with an AADL-like architectural syntax the Arcade approach allows models to be easily specified and efficiently analyzed.

Arcade models are built up out of components which are annotated with dependability information, as described in the AADL error annex. More advanced dependencies between components can be modeled using fault tree notation. Arcade also allows the modeling of repair modules with various strategies (such as dedicated repair or first come first serve).

Arcade uses so called deep compositionality, not only is the syntax compositional (i.e. models can be built by combining building blocks), but the semantics are also compositional. Each syntactical element has its own I/O-IMC behavior. The behavior of an entire model then emerges

as the composition of the component behaviors. The I/O-IMCs also allow 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 I/O-IMC model.

Details about this work are reported in [2, 3], we also report from the modelling process viewpoint in Section 2 of Deliverable D1.1.

Perspective The work on Arcade provides a compositional approach to giving semantics to an AADL-inspired modelling notation. We are going to look for synergies with the approach described in Section 2.1, and are continuing to develop tool support in this.

3 Extension of Statecharts

3.1 UPPAAL modelling and Concurrent Statecharts

Participants

- Kim Larsen, Alexandre David, and Arne Skou (AAU)

Contributions Timed Automata constitute one of the modelling formalisms that will be 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 considered:

- 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 it will be investigated whether the translation also can be applied for analyzing and testing the Terma industrial case.

3.2 Markov decision processes and STATEMATE

Participants

- Holger Hermanns, Reza Pulungan (SU)

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.

4 Library of component models

Work on this topic has not started yet. It is partly obsolete, due to changes in the consortium, discussed in Section 1.

5 Modelling tool improvements

Work on this topic is delayed, for reasons discussed in Section 1.

Bibliography

- [1] Eckard Böde, Marc Herbstritt, Holger Hermanns, Sven Jahr, Thomas Peikenkamp, Reza Pulungan, Ralf Wimmer, Bernd Becker. Compositional Reliability Evaluation for STATE-MATE. *IEEE Transactions on Software Engineering*, accepted for publication.
- [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] J.-P. Katoen, M. Bozzanol, G. Burte, A. Cimatti, M. le Coroller, V. Yen Nguyen, T. Noll and X. Olive, System and Software Co-Engineering: Performance and Verification, in: ESA ADCCS Workshop, Noordwijk, The Netherlands, 2008.