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

Deliverable no.: D5.4

Title of Deliverable: Plan for integration of tool components

Contractual Date of Delivery to the CEC:	**M12**
Actual Date of Delivery to the CEC:	**M13**
Organisation name of lead contractor for this deliverable:	**AAU**
Author(s):	**Kim G. Larsen, Alexandre David Holger Hermanns, Joost-Peter Katoen Brian Nielsen**
Participants(s):	**AAU CFV CNRS ESI RWTH SU**
Work package contributing to the deliverable:	**WP 1–4**
Nature:	**R**
Version:	**1.1**
Total number of pages:	**18**
Start date of project:	1 Jan. 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:

This deliverable describes the development of quantitative validations tools by individual partners within the two categories: tools for validation of probabilistic and stochastic systems, and tools for validation of real-time systems. Also, progress on specific tool components as well as experiements and plans for interaction between tools (including external and commercial tools) is described.

Keyword list: Probabilistic and stochastic tools, real-time tools, MRMC, PASS, MODEST, INFAMY, UPPAAL, UPPAAL-PROB, UPPAAL-TIGA, UPPAAL-TRON, UPPAAL-CORA, TORX, tool components, tool integration.

Contents

1 Introduction **4**

2 Tools for Probabilistic and Stochastic Analysis **5**
2.1 MRMC (RWTH) 5
2.2 PASS (SU) 5
2.3 MODEST (SU, ESI/Twente, RWTH) 6
2.4 INFAMY (SU) 7
2.5 UPPAAL-PROB (AAU) 7

3 Tools for Real-Time Analysis **8**
3.1 UPPAAL (AAU) 8
3.2 UPPAAL-TIGA (AAU, CFV) 9
3.3 UPPAAL-TRON (AAU) 9
3.4 UPPAAL-CORA (AAU, CNRS) 9
3.5 TORX (ESI/Twente) 10

4 Tool Components and Plans for Tool Integration **11**
4.1 Tool Components 11
4.2 Integration between Tools 13

Abbreviations

AAU: Aalborg University, DK

CFV: Centre Fèdèrè en Vèrification, B

CNRS: National Center for Scientific Research, FR

ESI: Embedded Systems Institute, NL

ESI/RU: Radboud University Nijmegen, NL

RWTH: RWTH Aachen University, D

SU: Saarland University, D

1 Introduction

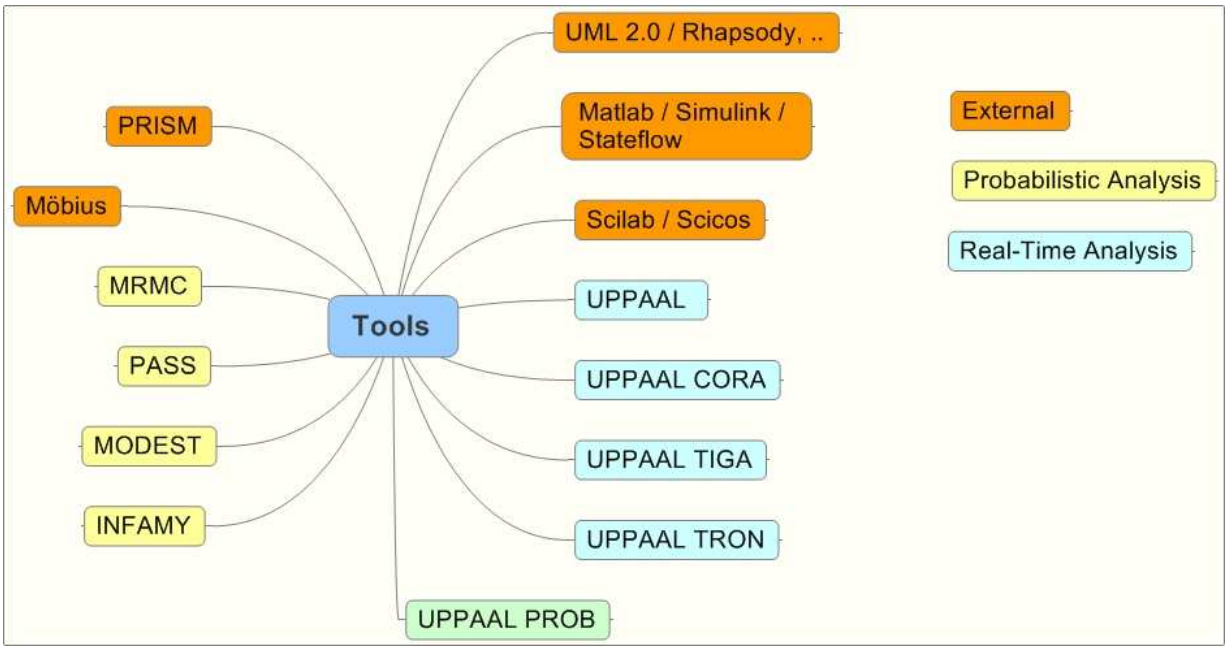

Figure 1: Internal and External Tools in Quasimodo

In this delieverable we describe the first year effort in Quasimodo on development of tools by individual partners (see Fig. 1), their availability as plug-in components and experiments and plans for exchange and interaction between internal tools and external (commercial) tools.

2 Tools for Probabilistic and Stochastic Analysis

2.1 MRMC (RWTH)

MRMC [1] is an open-source model checker for probabilistic models. It supports continuous-time and discrete-time Markov chains, reward extensions thereof, and uniform continuous-time Markov decision processes. Properties are expressed in stochastic variants of the branching-time temporal logic CTL. The key aspect are efficient algorithms for checking time-bounded reachability properties, i.e, the (maximal) probability to reach a certain set of goal states within a given deadline while avoiding to forbidden states. In the reward-based setting, in addition a cost-bound is imposed. Time-bounded properties of Markov chains of up to 10 million states can be checked in a matter of seconds. In addition to these reachability properties, long-run measures can be specified, as well as expected costs, and so on.

The core of the tool is based on numerical techniques to solve systems of linear equations together with advanced techniques to efficiently compute transient probabilities of CTMCs. In addition to the numerical engine, MRMC has been extended with means to check the validity of temporal formulas by means of discrete-event simulation. This work is motivated by the fact that previously known techniques based on hypothesis testing do neither support long-run measures nor unbounded reachability properties. Our approach is based on sequential confidence intervals and traditional discrete-event simulation of CTMCs, such as regenerative simulation.

In order to be able to reduce the state space of Markov (reward) chains, bisimulation minimization is supported. This facilitates the minimization of models prior to their analysis and has shown to yield drastic memory savings (up to exponential savings in state space size), as well as —and opposed to traditional model checking where mini- mization is more expensive than analysis— in verification time. The minimization can be parametrized with the property under consideration, yielding state spaces that are tailored to the measure-of-interest.

The command-line tool is used as a back-end model checker for stochastic Petri nets (in the tool GreatSPN), stochastic graph grammars, and more recently, discrete stochastic hybrid systems. It also interfaces to the PRISM model checker (at Oxford University) such that PRISM models can be directly fed into and analyzed (or minimized) by MRMC.

2.2 PASS (SU)

The probabilistic model checker PASS relies on predicate abstraction as its main vehicle to solve very large or infinite state Markov decision process models [28]. We have extended PASS to employ counterexample guided refinement now, leading to a genuine CEGAR approach for probabilistic models [16]. To mechanise this, we interfaced PASS to the SMT solver YICES [13], and the interpolant-generating theorem prover FOCI [25].

We have considered a selection of case studies including WLAN and sliding window protocols, to assess the performance of the tool, relative to the leading probabilistic model checker PRISM. The results are quite encouraging. To compete with PRISM on finite models, the benefit

[1] www.mrmc-tool.org/

of state space reduction has to offset the cost of repeating the CEGAR loop. On infinite or very large models only PASS can be used. We observe that for several models PASS is superior to the conventional approach, due to the significantly smaller abstract state space.

The approach is so far restricted, because it fails to provide lower probability bounds, it can only check if the reachability probability of a certain set of states is below a given threshold. In the future, we are planning to integrate ideas from [**?**] into PASS to overcome this shortcoming, and to make the power of PASS available to other tools developed in the consortium, see also below.

2.3 MODEST (SU, ESI/Twente, RWTH)

MODEST [2] specifications constitute a coherent starting-point to analyse distinct system characteristics with various techniques, e.g., model checking to assess functional correctness and discrete-event simulation to establish the system's reliability. Analysis results thus refer to the same system specification, rather than to different (and potentially incompatible) specifications of system perspectives like in UML. With MODEST we take a single formalism, multisolution approach.

The formalism of probablistic timed automata (PTA) combines discrete probabilities and nondeterministic time. We are developing a model checker tool for PTA specified in the language MODEST. This language includes features such as exception handling, dynamic parallelism and recursion. A large subset of MODEST maps to PTA.

This tool does not use purely symbolic techniques relying on regions or zones to represent continuous time, as done in the work on UPPAAL Pro [put pointer]. Instead we experiment with an integral semantics [1] which relies on (bounded) integer variables to represent clocks. A considerable set of interesting properties is preserved under this semantics. Our tool first parses a MODEST model and then extracts the parallel components occuring therein. It then generates the underlying PTA for each component, and translates each into a set of (purely probabilistic) guarded command modules using the integral representation of time. The output format is that of the leading probabilistic model checker PRISM, also accepted by our abstraction-refinement based model checker PASS. In this way the tool can be seen as a preprocessor, since the genuine verification task is then delegated to a backend probabilistic model checker.

We have chosen to use C# to implement the tool, including the C#-based GPPG/GPLEX tools for the parsing stage, significantly improving cross-platform compatibility and deployment compared to the existing, C++-based MODEST simulation environment MOTOR.

As a testbench, the tool has been applied to the BRP (bounded retransmission protocol) example using PRISM as the backend probabilistic model checker. Our goal is to support other backend tools as well, especially PASS, in the close future, and also explore plug-in possibilities into UPPAAL Pro. However, for this we have to overcome a technical problem, rooted in the fact that the synchronization disciplines supported by UPPAAL differ semantically from the one used in MODEST, PRISM and PASS, which so far causes all straightforward translation attempts to fail.

[2] fmt.cs.utwente.nl/tools/motor/

2.4 INFAMY (SU)

INFAMY is a new model checker for the analysis of infinite-state continuous time Markov models. It implements the theory developed in [29]. The model structure is not restricted to adhere a certain regular structure. The general idea behind INFAMY is to explore on-the-fly only that part of the model that is neccessary to refute or validate a property, up to prespecified error probability. The concept has shown benefits on a number of case studies. These include models from network performance estimation as well as queueing models and models based on biological systems. As finding the optimal truncation points is often much too costly, `INFAMY` provides a set of different algorithms to find a good approximation of the minimal depth to explore. This allows for a trade-off between the memory used to store the model and the time needed to find this point.

INFAMY includes a number of advanced features. Its input models can be given in a guarded command language very close to that of the PRISM model checker, but including extensions for describing variables with infinite range, as in PASS. Properties are specified in a rich subset of CSL.

INFAMY is implemented in a modular way using C++. PRISM models are translated into C code and compiled to allow for a high-speed construction of the truncated model. The modular structure of the tool and the usage of templates also allows for the implementation of several representationd of the finite truncation of the model in memory. We have implemented a sparse matrix representation as well as an implicit representation where only the states but no transitions have to be stored in memory, as these are automatically re-generated each time a state is visited.

As future extensions we are mainly focussing on improving the model exploration to allow for smaller memory usage and run time. Additionally, we are going to improve the integration of the tool with existing Markov model checkers. In particular,, we are planning to link this tool to the MRMC model checker for CSL developed in the consortium.

2.5 UPPAAL-PROB (AAU)

UPPAAL-PROB is the most recent branch of UPPAAL offering for the first time a tool that allows simultaneous analysis of probabilistic and dense real-time properties. The underlying modeling formalism is that of probabilistic timed automata [15, 22, 21].

Following the efficient on-the-fly exploration paradigm applied in all branches of UPPAAL's, the implementation in UPPAAL-PROB applies forward state space exploration of the underlying timed automata. The forward exploration is – as in UPPAAL TIGA – combined with backwards stabalization aiming at producing a coarse(st) finite partitioning of the state space. The final partitioning has the properties of a (probabilistic) time-abstracted bisimulation and constitutes a finite-state Markov decision process based on which the desired infimum and supremum probabilities can be obtained using linear programming. Currently, LP-SOLVE[3] is used as back-end, but during next year the usage of PRISM and MRMC as back-ends will be explored.

A first public release of the tool will be available by mid-February 2009.

[3] http://www.cs.sunysb.edu/ algorith/implement/lpsolve/implement.shtml

3 Tools for Real-Time Analysis

3.1 UPPAAL (AAU)

UPPAAL [4] is an integrated tool environment for modeling, validation, performance evaluation, scheduling, testing and controller synthesis for real-time systems modeled as networks of timed automata [1], extended with discrete variables, user-defined data-types and functions. The tool-suit features an integrated editor, a simulator, and a verifier. Whereas the classical version of UPPAAL is targeted towards verification based on symbolic model-checking, the tool-suit contains branches target to different application. Among the branches,

- UPPAAL-CORA supports efficient cost-optimal reachability for priced timed automata and has been extensively for optimal planning and scheduling
- UPPAAL-TIGA is based on timed game automata and its purpose is code synthesis,
- UPPAAL-TRON applies the symbolic model-checking engine of UPPAAL to on-line conformance testing of real-time systems,
- UPPAAL-PROB is the most recent branch of UPPAAL supporting probabilistic reachability of probabilistic timed automata

The design of the tool-suit is based on a client-server architecture where the graphical user interface (GUI) communicates with the model-checker (engine). This makes it easier to integrate with other tools that could use the same communication protocol. Furthermore, the file format we are using is XML, which is a plain text open format that can be read or written by other tools.

The current extensions on the UPPAAL tool include a concrete simulator, stop-watches, and robust reachability analysis.

The idea of the *concrete simulator* is to visualize states with their clock valuations, in contrast with the current symbolic view. This is more intuitive for users and allows us to offer new functionalities such as a Ghantt chart to visualize different predicates in function of the time. So far only UPPAAL-TIGA is equipped with such a simulator for playing strategies and we want to integrate it to UPPAAL with the capability of generating and playing traces.

The extension of *stop-watches* implements an efficient over-approximation technique to deal with the added functionality of stopping any clock on any state. The exact analysis is undecidable. One useful application is schedulability analysis [11] where we believe that for the models we are considering the analysis is in fact exact. Stop-watches can also be used to encode more general linear-hybrid automata [7] and we can use efficient techniques based on timed automata to analyse them.

The last extension concerns *robust reachability analysis*. The standard analysis techniques we use assume that all the clocks are perfectectly synchronous and never drift. This is not the case in reality and we have implemented a reachability algorithm that let clocks drift [8].. The technique is based on detecting and stabilizing loops with respect to clock drifts [12].

[4] www.uppaal.com

3.2 UPPAAL-TIGA (AAU, CFV)

UPPAAL-TIGA implements the first efficient on-the-fly algorithm for solving games based on timed game automa with respect to both safety and reachability control objectives. In the newest distributed version of UPPAAL-TIGA, support for synthesizing *realizable* strategies is available. In particular, synthesis of strategies with only partial observability is supported and synthesis of safety strategies ensuring zoneness may be obtained as a special case of general support for synthesis wrt. Büchi objectives.

In a previous case study [19] we connected UPPAAL-TIGA to simulink in a tool chain to generate code from a model description in the formalism of timed game automata. The connection was done via ad-hoc scripts written for this particular case-study. The Hydac case-study proved to be different w.r.t. code generation. From this experience we plan to define a class of timed game automata that can be used for generating code automatically in a common framework. As far as the connection with Simulink is concerned, this means a generic way to integrate the generated code as an S-function with its inputs and outputs connected to the rest of a Simulink model.

In addition, we are extending the development of UPPAAL-TIGA with partial observability. The implementation of the new algorithm presented in [5] is being finalized inside UPPAAL. The new implementation takes advantages of optimizations in the model-checker and extends the GUI. The model of timed games with partial observability may be better suited for code generation and connection to other tool in the sense that the strategy produced is an automaton that depending on observed inputs or sensed values takes necessary actions until its observation changes.

The plan for UPPAAL-TIGA is to complete an automatic tool-chain from the model to executable code. Going through Simulink is an important step since we can simulate the code in a more advanced manner.

3.3 UPPAAL-TRON (AAU)

Testing with UPPAAL-TRON is limited to what we can model in timed automata. Connecting to Simulink would improve testing with added quantitative aspects. We plan to make this connection in such a way that they would become one tester component. This component would communicate with the implementation under test considered as a black box (as UPPAAL-TRON is doing today). The connection we are investigating could be done either by including UPPAAL-TRON inside Simulink as an S-function or having UPPAAL-TRON communicate with Simulink via an adapter API.

The plan for this integration is to improve UPPAAL-TRON capabilities but also to add the power of automatic testing for people who are using Simulink to simulate their systems.

3.4 UPPAAL-CORA (AAU, CNRS)

UPPAAL-CORA offers support for efficient cost-optimal reachability for priced timed automata. A generalization of the forward symbolic reachality algorithm of UPPAAL is obtained using

a notion of priced zone – zones extended with affine cost function – permitting pruning and guiding of the search.

Prototype versions of UPPAAL-CORA using agent-based search engines [27] and direct model checking using new heuristic functions for guiding the search (similar to the Russian Doll principle) [20] it is possible to find near-optimal solutions very efficiently. During the next period effort towards enabling these prototypes implementation for the full modeling formalism of UPPAAL-CORA will be pursued.

Also, implementation of support for optimal infinite scheduling [4, 14] and optimal reachability for multi-priced timed automata [23] will be pursued.

3.5 TORX (ESI/Twente)

A new version of the on-line testing tool TORX [5], JTORX[6], is being developed. The main focus at the moment is ease of deployment, e.g. to quickly demonstrate the idea of model-based testing, and for student exercises.

Support for basic LTS formalisms (CADP .aut files, graphml) is built in. In addition, it is able to use the modelling-language specific LTS-access components of TorX, as long as the labels do not contain variables. This can be used to access e.g. LOTOS or mucrl or mcrl2 models. For the moment JTorX is lacking support for symbolic or time features, with the consequence that neither the timed extension of TorX, nor the Promela support in TorX can be used in JTorX. JTorX has support for the TorX Adapter interface.

Currently there are no plans to integrate with UPPAAL. This is due to lack of need for the basic features that we are developing now. If integration with UPPAAL might make our life easier by the time that we start work on the symbolic/timed extensions, it will be taken into consideration – certainly usage of the DBM-library of UPPAAL will be made.

[5] fmt.cs.utwente.nl/tools/torx

[6] fmt.cs.utwente.nl/tools/jtorx/

4 Tool Components and Plans for Tool Integration

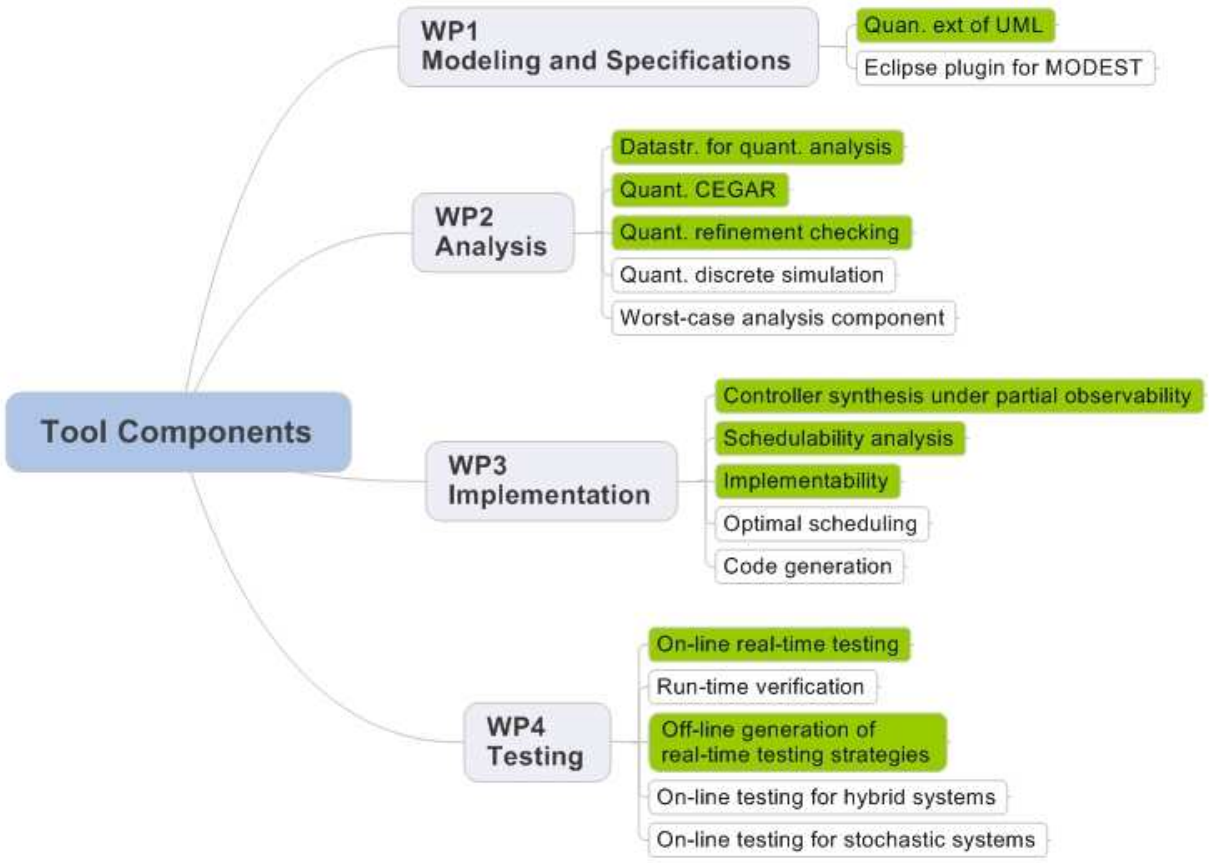

Figure 2: Quantitative Tool Components to be developed within Quasimodo. Progress has been made on highlighted components.

4.1 Tool Components

Quasimodo will contribute with a number of unique tool components for quantitative analysis and validation of aspects relevant for the various phases of the development proces ranging from requirement capturing and design to implementation and testing. The Quasimodo tool components will – as detailed in Section 4.2 be made available as plug-ins allowing for integration with existing commercial embedded tools, e.g. SimuLink, Rhasody, Scilab/Scicot.

The development of tool components invisaged in the Quasimodo DoW (Description of Work) is given in Fig. 2. As indicated, progress has been made on several of these components. Below we provide more details on this:

Quatitative extensions of UML the translation of XMI format for UML Statechart into UPPAAL's XML format offers the possibility of supporting analysis of real-time profiles of UML. In

particular it has been agreed to meet with Sebastian Gerard (CEA) in order to investigate support of the MARTE UML profiles and possibly interaction with the Papyrus implementation of this profile [7].

Datastructures and Algorithms: a number of datastructures and algorithms for quantitative state space representation and manipulation have been developed during the first year, e.g. [20].

CEGAR: Support for counter-example guided abstraction/refinement has been made in a probabilistic setting [17].

Quantitative refinement checking: Algorithmic support for probabilistic and stochastic simulation has been given and evaluated in [2, 30]. In the setting of timed automata, encodings of a variety of refinements into timed game problems has been given in [3].

Controller synthesis under partial observability: An implementation in UPPAAL-TIGA of the algorithm in [5] for synthesis of control strategies under partial observabilities has been made in [9].

Schedulability analysis A modelling framework for schedulability analysis of embedded applications on multi-processor architectures has been made in [11]. The framework includes a rich collection of attributes for task (best- and worst-case execution times, minimum arrival times, deadlines and priorities), dependencies between tasks, assignment of resources, a variety of scheduling policies (including FIFO, EDTa dn FPS), and possibilities of preemption.

Implementability checking: In [8] an implementation in UPPAAL of the algorithm for robustness analysis of timed automata from [12] has been made. That is, the correctness of the model is analysed in the presence of small drifts on the clocks.

Optimal scheduling: Optimal scheduling has been performed substantially using UPPAAL-CORA. In particular, the application of agent-based search engines in [27] has proven particular succesfull. In a similar manner [20] provides heuristic-guided search which leads to designated goal-states substantially faster than the standard UPPAAL verification engine.

On-line real-time testing: Both UPPAAL-TRON and TORX provide algorithmic support for online real-time testing [18].

Off-line real-time testing Using UPPAAL and UPPAAL-TIGA off-line generation of strategies for conformance testing of real-time systems has been pursued for increasingly expressive classes of timed automata specifications [10, 24].

[7] www.papyrusuml.org

4.2 Integration between Tools

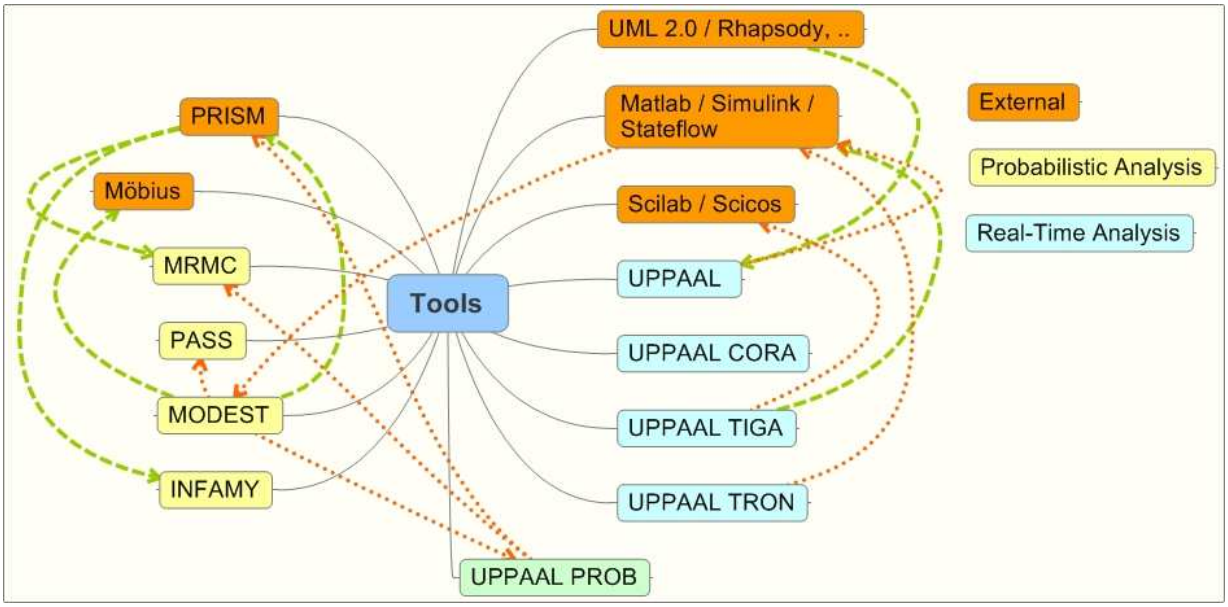

Figure 3: Progress on and plans for integration between tools

In this section we report on the progress and plans within the Quasimodo project for interaction and integration between tools of the consortium and between tools of the consortium and external/commercial tools, e.g. PRISM, Möbius, Matlab/Simulink, Scilab/Scicot, and UML/Rhapsody.

Probabilistic Tools

PRISM [8] is a leading (external) tool for model checking and validation of probabilistic systems with numereous applications ranging from correctness of security protocols, efficiency of wireless protocols, reliability of nanotechnology designs, to the analysis of signalling pathways.

Obviously, a number of links for interaction between PRISM and Quasimodo tools have been made or are under investigation:

MODEST: The modeling formalism of MODEST is extremely expressive and translations from models in (a subset of) MODEST to PRISM. In a similar manner MODEST models may be translated to Möbius allowing for performance analysis using simulation.

MRMC *and* INFAMY: Interfaces between PRISM and MRMC as well as INFAMY are made allowing PRISM models to be fed into and analyzed by the novel engines of these tools.

[8] http://www.prismmodelchecker.org/

UPPAAL-PROB*:* UPPAAL-PROB computes – in an abstraction/refinement manner – stable partitioning of the state-space of probabilistic timed automata. Future interaction with PRISM and MRMC should allow the analysis of the resulting finite Markov decision processes to be performed by the exiting engines of these tools. Also, translations of PRISM and MODEST models of (discrete-time) probabilistic timed automata into UPPAAL-PROB will be pursued. Here a main challenge will be how to deal with the differences in synchronization mechanisms.

Finally, the use of probability distributions for modeling noice in Simulink (as used in the performance evaluation of the HYDAC case [6]) will be compared with the formal probabilistic and stochastic modeling formalisms pursued with Quasimodo (here represented by MODEST). This comparison is a prerequisite for applying the probablistic tool components of Quasimodo as plug-ins to Matlab/Simulink.

Real-time Tools

Matlab/Simulink being the defacto tool used by European industries a main goal of Quasimodo is to seek integration with this tool to increase impact. Dispite the leave of the industrial partner Incron Gmbh, and unsuccessful attempt of direct collaboration with Mathworks, Quasimodo pursue this direction fully and to provide the following links to Matlab/Simulink (we refer to more details in the Project Periodic Report for the first year period):

Controller Synthesis: We will continue the successful work in [6] to systematize how the control-strategies generated by UPPAAL-TIGA can be represented as Simulink S-functions allowing it to be analysed in context of continous behavior by simulation, and enabling code to be generated to a target platform supported by Simulink.

Quantitative models: We will establish a link between the quantitative automata based modelling notations used in the Quasimodo consortium and its tool components and the ones used in Simulink (esp. Stateflow). Having this link will make it easer to migrate models used by Quasimodo developed tool components with Simulink. We foresee that this link will be established in part by describing how the commonly used formalisms of timed automata (TA), and its linear hybrid (LH-TA) and hybrid (H-TA) extensions can be encoded as Simulink models, and in part by studying the feasibility of extending the Simulink modelling framework with similar quantitative syntactic and semantic extensions that Quasimodo is researching for UML.

Model-Checking: The powerfull real-time model-checking technology of UPPAAL will be available to Stateflow models by a) providing a translation of a subset of uppaal-timed automata to Stateflow models, or b) by creating a UPPAAL S-Function. This will allow model-checked component models to be simulated in context of existing or more detailed (e.g., with continous behavior) Simulink model-components to evaluate the combined system level behavior.

Refinement Checking by Testing: Although the UPPAAL model-checker is powerfull it happens that the developed models are beyond what UPPAAL can analyze, e.g., if the model is too large (in terms of number of concurrent components or clocks), is too concrete (contains too many details and data-space), or contains continous behavior described by differential equations. However, it is possible to partially check that such detailed models refines a more abstract one (that may have been model-checked) by means of testing, i.e., a simulation of the detailed model will be used as implementation under test.

Quasimodo will develop two tool components by extending UPPAAL-TRON (and its underlying rt-ioco testing refinement relation). A first goal is facilitate refinement testing of large timed automata models. A second and more ambitious goal is to test whether a Matlab/Simulink model, possibly with continous behavior, conforms (refines) a more abstract timed automata model.

This second integration effort will take place in synergy with the MULTIFORM project (EU-FP7-224249). Quasimodo will emphasize development of the semantic foundation and making UPPAAL-TRON integratable, where as MULTIFORM will emphasize strong tool integration and demonstration.

Scilab

At the same time we will evaluate the possibilities and benefits of similar integrations with Scilab[9] – for more details see Project Periodic Report.

UML State-Charts

Finally, AAU has implemented a translation from UML state-charts to timed automata involving the tools Rational Systems Developer and UPPAAL. The translation is done by first exporting the UML state-charts into the standard XMI format, and then translating into the UPPAAL XML format. The translation will permit verification as well as automatic test generation from state-chart models, and is foreseen to be applied in the industrial case-study of Terma.

[9] http://www.scilab.org/

References

[1] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2), 1994.

[2] Jonathan Bogdoll, Holger Hermanns, and Lijun Zhang. An experimental evaluation of probabilistic simulation. In *28th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE)*, volume 5048 of *Lecture Notes in Computer Science*, pages 37–52. Springer, 2008.

[3] Peter Boulychev, Tomas Chatain, Alexandre David, and Kim G. Larsen. Playing games with timed games. Under submission, 2009.

[4] Patricia Bouyer, Ed Brinksma, and Kim G. Larsen. Optimal infinite scheduling for multi-priced timed automata. *Formal Methods in System Design*, 32(1):2–23, February 2008.

[5] Franck Cassez, Alexandre David, Kim Guldstrand Larsen, Didier Lime, and Jean-François Raskin. Timed control with observation based and stuttering invariant strategies. In Kedar S. Namjoshi, Tomohiro Yoneda, Teruo Higashino, and Yoshio Okamura, editors, *ATVA*, volume 4762 of *Lecture Notes in Computer Science*, pages 192–206. Springer, 2007.

[6] Franck Cassez, J. J. Jessen, Kim G. Larsen, Jean-François Raskin, and Pierre-Alain Reynier. Robust and optimal contorllers - an industrial case study. In *To appear in Proceedings of HSCC'09*, 2009.

[7] Franck Cassez and Kim Guldstrand Larsen. The impressive power of stopwatches. In Palamidessi [26], pages 138–152.

[8] Alexandre David, Piotr Kordy, Kim G. Larsen, and Jan Willen Polderman. Practical robustness analysis of timed automata. Under submission, 2008.

[9] Alexandre David, Kim G. Larsen, and Didier Lime. Uppaal-tiga 2009: Towards realizable strategies. Under submission, 2009.

[10] Alexandre David, Shuhao Li, Brian Nielsen, and Kim G. Larsen. A game-theoretic approach to real-time system testing. In *DATE*, pages 486–491. IEEE, 2008.

[11] Alexandre David, Jacob I. Rasmussen, Kim G. Larsen, and Arne Skou. *Model-based Framework for Schedulability Analysis Using UPPAAL 4.1*. Taylor ad Francis, 2009. To appear in CRC Press Book on "Model-Based Design of Heterogeneous Embedded Systems".

[12] Conrado Daws and Piotr Kordy. Symbolic robustness analysis of timed automata. In Eugene Asarin and Patricia Bouyer, editors, *FORMATS*, volume 4202 of *Lecture Notes in Computer Science*, pages 143–155. Springer, 2006.

[13] B. Dutertre and L. M. De Moura. A fast linear-arithmic solver for dpll. In *In Proceedings of CAV 2006*, 2006.

[14] Ulrich Fahrenberg and Kim G. Larsen. Discount-optimal infinite runs in priced timed automata. In *Proceedings of INFINITY 2008 10th International Workshop on Verification of Infinite-State Systems*, 2008.

[15] H. Gregersen and H. E. Jensen. Formal design of reliable real time systems. Master's thesis, Department of Matematics and Computer Science, Aalborg University, Denmark, 1995.

[16] H. Hermanns, B. Wachter, and L. Zhang. Probabilistic cegar. In *In Procedings of CAV 2008*, 2008.

[17] Holger Hermanns, Björn Wachter, and Lijun Zhang. Probabilistic CEGAR. In Aarti Gupta and Sharad Malik, editors, *20th International Conference on Computer Aided Verification (CAV)*, volume 5123 of *LNCS*, pages 162–175. Springer, 2008.

[18] Anders Hessel, Marius Mikucionis, Brian Nielsen, Paul Pettersson, Arne Skou, and Kim G. Larsen. *Testing Real-Time Systems Using UPPAAL*, volume 4949 of *LNCS*. 2008.

[19] Jan Jakob Jessen, Jacob Illum Rasmussen, Kim Guldstrand Larsen, and Alexandre David. Guided controller synthesis for climate controller using uppaal tiga. In Jean-François Raskin and P. S. Thiagarajan, editors, *FORMATS*, volume 4763 of *Lecture Notes in Computer Science*, pages 227–240. Springer, 2007.

[20] Sebastian Kupferschmid, Jörg Hoffmann, and Kim G. Larsen. Fast directed model checking via russian doll abstraction. In *Proceedings of TACAS 2008*, volume 4963 of *Lecture Notes in Computer Science*, 2008.

[21] Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Jeremy Sproston. Performance analysis of probabilistic timed automata using digital clocks. *Formal Methods in System Design*, 29(1):33–78, 2006.

[22] Marta Z. Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Verifying quantitative properties of continuous probabilistic timed automata. In Palamidessi [26], pages 123–137.

[23] Kim G. Larsen and Jacob I. Rasmussen. Optimal reachability for multi-priced timed automata. *Theoretical Computer Science*, 390(2-3):197–213, 2008.

[24] Shuhao Li, Alexandre David, Kim G. Larsen, and Brian Nielsen. Cooperative testing of uncontrollable timed systems. In *Fourth Workshop on Model-Based Testing (MBT'08)*, March 2008.

[25] K. L. McMillan. an interpolating theorem prover. *Theoretical Computer Science*, 345, 2005.

[26] Catuscia Palamidessi, editor. *CONCUR 2000 - Concurrency Theory, 11th International Conference, University Park, PA, USA, August 22-25, 2000, Proceedings*, volume 1877 of *Lecture Notes in Computer Science*. Springer, 2000.

[27] Jacob Illum Rasmussen, Gerd Behrmann, and Kim Guldstrand Larsen. Complexity in simplicity: Flexible agent-based state space exploration. In Orna Grumberg and Michael Huth, editors, *TACAS*, volume 4424 of *Lecture Notes in Computer Science*, pages 231–245. Springer, 2007.

[28] B. Wachter, L. Zhang, and H. Hermanns. Probabilistic model checking modulo theories. In *Proceedings of QEST 2007*, 2007.

[29] L. Zhang, H. Hermanns, E. M. Hahn, and B. Wachter. Time-bounded model checking of infinite-state continuous-time markov chains. In *In proceedings of ACSD 2008*, 2008.

[30] Lijun Zhang, Holger Hermanns, Friedrich Eisenbrand, and David N. Jansen. Flow faster: Efficient decision algorithms for probabilistic simulations. *Special Issue on TACAS 2007, Logical Method in Computer Science (LMCS)*, 2008.