Abstract: Time4Sys in a nutshell

(Tool presentation)

Loïc Fejoz
RealTime-at-Work
Nancy, France
Email: loic.fejoz@realtimeatwork.com

Yassine Ouhammou
LIAS/ISAE-ENSMA
Poitiers, France
Email: yassine.ouhammou@ensma.fr

***Abstract*—Waruna project is a collaboration between academic and industrial actors funded through the french government grant called FUI (Fond Unique Interministriel). The underlying idea behind this four-years project (2015-2019) is that the temporal performance verification phase of real-time systems development life-cycle shall not be a barrier for non-experts of the domain. Hence, the main objective of Waruna project is to ease the integration of the temporal performance verification in engineering practices. Time4Sys is the framework derived from the Waruna project and whose development is in progress. As an integrated framework aiming to fill the gap between system engineering design models and timing models, Time4Sys is based on two components. The front-end which is dedicated to modellers in order to design real-time systems using a graphical language. The back-end shall be a customized part which allows modellers to analyze iteratively and accurately their designs with different analysis tools. The customization of the back-end consists of managing the transformation between designs and analysis tools and the orchestration of tests that correspond to the designs under-analysis. In other words, the flexibility of the back-end part enables to capitalize the experience of temporal verification analysts as a referential followed by modellers via the front-end without being obliged to have a deep knowledge of the temporal verification. Time4Sys is based on model-driven engineering settings and developed as a Polarsys plugin. Modeling, traceability, transformation, analysis, and result reporting activities are the pillars of the Time4Sys structure and are explicitly formalized as a set of meta-models.**

I. Context

The development of safety-critical real-time embedded systems (RETS) requires analyzing functional and non-functional requirements in order to check if the design meets these requirements including temporal requirements. For the latter, a plethora of analysis tests have been developed by the real-time community leading to a set of academic and industrial analysis tools (like MAST [1], Cheddar [2] and RTaW-Pegase [3]). Model-driven engineering (MDE) [4] paradigm gains in term of popularity and become widely used by practitioners. Indeed, while the design of RTES is expressed in architectural design languages (like AADL [5], SysML [6] and UML-MARTE [7]) and the temporal verification is performed by analysis tools, MDE allows architects (designers and analysts) to translate their architectures from a formalism to another and also integrate analysis phase at an early design stage to avoid errors that can impact sharply the development life-cycle and the time-to-market.

II. Problem statement

Due to the criticality of RTES, the timing verification should be as accurate as possible. The timing verification is still driven by analysts experience. Indeed, the current development of RTES requires several come-and-go flows between designers and expert analysts. However, many projects can not afford experts to help for analysis. Even when such help exist, it is hard to know which toolbox to apply on the problem and the conditions of application. Even with experts available, the automation, or at least the guidance, of timing verification is hard because of the semantic gaps between systems engineering design artifacts and timing models of such systems. The Time4Sys framework aims to fill this semantic gap.

III. Background

The Waruna project was born from the needs of an SME (Small and Medium-sized Enterprise) and a big OEM (Original Equipment Manufacturer) both building critical embedded systems with hard real-time constraints. They have quite different levels of real-time expertise. Still, a common solution can helps all project's partners. The objective is two-fold. First is the integration of the timing verification transformation closely to the engineering software tool, so that its semantic can be fully used and translated to verification tools. The second is to reuse and aggregate all knowledge from previous projects presented hereafter.

- UML-MARTE is an OMG standard design language [7]. Since it is dedicated to RTES, Time4Sys reuse most of its relevant architectural and some temporal properties.
- TEMPO [8] is a MARTE viewpoint developed as an intermediate model in order to reduce and bridge the semantic gap between MARTE models and the analysis tools models. Hence, instead of adapting MARTE models to fit with an analysis tool, TEMPO suggests adding an intermediate step to explicit the adaptation.
- MoSaRT is a framework [9] based on two pillars: a design language and analysis repository. Contrary to MARTE, MoSaRT language is dedicated to express timing analysis models. The repository enables analysts to share their experiences by describing the real-time contexts as a set of axioms and their appropriate tests. The content of the repository makes the designer autonomous during the analysis phase since it can be used as a decision-support.

Some of the temporal features and the repository side of MoSaRT shall be integrated into Time4Sys.

- LTTng [10] is a project that enables the exchange and analysis of kernel traces. Then it has been extended to other kinds of traces. Its meta-model is known to be efficient to handle. Thus Time4Sys has also adapted its core.

IV. Time4Sys Foundations

By using MDE settings, Time4Sys is being developed as an Eclipse Polarsys plugin. Time4Sys proposes four capabilities: the design, the analysis, the traceability and the reporting.

1) Design. Time4Sys provides a graphical domain-specific language to design RTES. While the concrete syntax is based on Capella [11], the abstract syntax is a meta-model based mainly on an extraction of MARTE and some MoSaRT concepts. This meta-model is expressed in Ecore [12].
2) Traceability. Because of the previously mentioned semantic gap, timing design models are transformed to analysis model. We assume that analysis models are expressed in the same meta-model as design models, but with more restrictions. Unfortunately, those restrictions depend on the targeted analysis and verification tools. Moreover, because we do not want experts to be involved too much, verification results must be brought back to the user interface. So full traceability of those transformations is mandatory. Here again, Time4Sys traceability meta-model (called mapping meta-model) is an aggregation of the trace mechanisms of QVT [13] and ATL [14].
3) Analysis. Time4Sys contains several connections allowing it to be used with different tools like MAST and RTaW-Pegase++™. This latter provides a configurable simulation with results deeply integrated into the Time4Sys framework.
4) Result reporting. The underlying idea is to present results to users after analysis. Once the results are validated by users, new models integrating results can be generated. Currently, we start examining the trace as a result of simulation tests. So Time4Sys is able to handle trace. This capability is based on a meta-model inspired from Trace Compass [15], LTTng and Complex Event Processing. Indeed it is mandatory to analyze traces from different viewpoints (e.g., the hardware resources usage, the task's activations, etc.). All those views form a kind of lattice which is a unique feature of this meta-model.

V. Proof of concept

While the tool is not yet available for download[1], it is already submitted to the Polarsys [16] consortium so as to be shared with an Open Source License (namely the EPL).

[1]Note for reviewers: it will be available by the time of the conference

VI. Conclusion and future works

Early experiments are very promising and most foreseen use-cases can be modeled with Time4Sys design language. Their analyses are available.

In the future, it is planned to provide an advanced constraints meta-model so that systems engineers could express their requirements. It is also planned to provide connections to other tools (like Simso [17] and CPAL [18]).

Acknowledgment

The authors would like to thank all the partners of the project.

This project is supported by the French ministry of the Economy, Finances, and Industry. This project is co-funded by the following Regions: Île-de-France, Occitanie, Grand-Est, and Grand-Nancy. This project is co-funded by the European Union. Europe is committed in France through the European Funds for Regional Development and the European Social Fund.

References

[1] MAST, "Modeling and analysis suite for real-time applications," http://mast.unican.es/, [Last access 14/04/2017].
[2] "The cheddar project : a gpl real-time scheduling analyzer," http://beru.univ-brest.fr/ singhoff/cheddar/, [Last access 14/04/2017].
[3] RTaW-Pegase, "Modeling, simulation, and timing analysis for communication networks," http://www.realtimeatwork.com/software/rtaw-pegase/, [Last access 14/04/2017].
[4] K. Balasubramanian, A. S. Krishna, E. Turkay, J. Balasubramanian, J. Parsons, A. S. Gokhale, and D. C. Schmidt, "Applying model-driven development to distributed real-time and embedded avionics systems," *IJES*, vol. 2, no. 3/4, pp. 142–155, 2006.
[5] AADL, "Architecture analysis and design language," http://www.aadl.info/aadl/currentsite/.
[6] SysML, "Systems modeling language," http://www.omg.org/spec/SysML, [Last access 14/04/2017].
[7] MARTE, "Modeling and analysis of real-time and embedded systems," http://www.omg.org/omgmarte/, [Last access 14/04/2017].
[8] R. Henia, L. Rioux, and N. Sordon, "Demo abstract: TEMPO: integrating scheduling analysis in the industrial design practices," in *IEEE RTAS*, 2016, p. 63.
[9] Y. Ouhammou, E. Grolleau, M. Richard, P. Richard, and F. Madiot, "Mosart framework: A collaborative tool for modeling and analyzing embedded real-time systems," in *CSDM*, 2014, pp. 283–295.
[10] LTTng, "an open source tracing framework for linux," http://lttng.org/, [Last access 17/04/2017].
[11] "Capella," https://polarsys.org/capella/, [Last access 14/04/2017].
[12] "Eclipse modeling framework," https://www.eclipse.org/modeling/emf/, [Last access 14/04/2017].
[13] "Query/view/transformation QVT," http://www.omg.org/spec/QVT/, [Last access 14/04/2017].
[14] F. Jouault and I. Kurtev, "Transforming models with ATL," in *Satellite Events at the MoDELS 2005 Conference*, 2005, pp. 128–138.
[15] "Eclipse trace compass," [Last access 14/04/2017]. [Online]. Available: http://tracecompass.org
[16] "Time4sys," [Last access 14/04/2017]. [Online]. Available: https://www.polarsys.org/projects/polarsys.time4sys
[17] Simso, "Simulation of multiprocessor scheduling with overheads," http://projects.laas.fr/simso/, [Last access 14/04/2017].
[18] L. Fejoz, N. Navet, S. M. Sundharam, and S. Altmeyer, "Demo abstract: Applications of the CPAL language to model, simulate and program cyber-physical systems," in *IEEE RTAS*, 2016, p. 64.