Electronic Communications of the EASST
Volume X (2013)

Proceedings of the Automated Verification of Critical Systems (AVoCS 2013)

Simulating Timed UML2 Sequence Diagrams with Timed CSP

Alexander Knapp, Liam O'Reilly, Markus Roggenbach, Bernd-Holger-Schlingloff

3 pages

Guest Editors: Steve Schneider, Helen Treharne
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Simulating Timed UML2 Sequence Diagrams with Timed CSP

Alexander Knapp[1], **Liam O'Reilly**[2], **Markus Roggenbach**[2], **Bernd-Holger-Schlingloff**[3]

[1]Universität Augsburg, [2]University of Swansea,
[3]Fraunhofer FOKUS and Humbodt Universität zu Berlin

Abstract: This paper deals with the formal validation of requirements in a model-based design methodology. We consider timed UML2 sequence diagrams as formalization of informal requirements and show how to simulate them by translation into Timed CSP. Simulation results are displayed in a graphical representation. This way, we can detect errors in the requirements at a very early stage in the development. We illustrate our ideas with an example from a ventricular assist device, which is a highly safety-critical application for supporting the human heart.

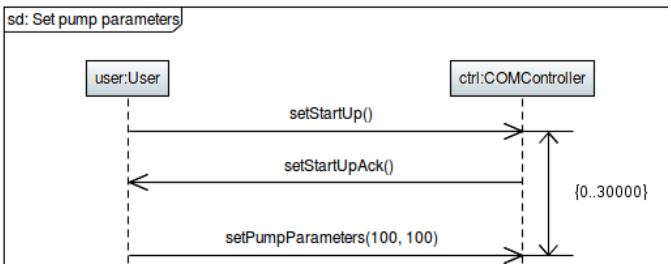
Keywords: Validation, Simulation, UML2, Sequence Diagrams, Timed CSP

1 Introduction

Model-based engineering is a current paradigm for the development of complex software systems. Starting from informal requirements, the structure and behaviour of the system under construction is modelled with various formalisms. In this context, *UML2 sequence diagrams* are frequently used for exhibiting sample runs during the requirements elicitation phase. Often, time is an integral aspect of such requirements. In UML2, real-time constraints can be expressed by timers, as well as minimal and maximal waiting times between certain event occurrences. In requirements validation, catching timing errors plays an important role. The interplay of different timing constraints in a sequence diagram can be confusing. Here, *simulation* of sequence diagrams can be helpful: as an interactive validation method, where the user provides a timed run which the simulator accepts or rejects; or as a means against "organizational blindness", where the simulator automatically generates random runs – which might be of a form beyond the modeller's imagination.

2 A Ventricular Assist Device

As an example, we consider a ventricular assist device for the support of the human heart. This example is adapted from an industrial project, which is currently under development. A typical use case is that the user (i.e., the doctor or nurse) wants to adjust some parameters from a panel PC via a serial connection. Here, the user is restricted to a 30 second time window to input the new configuration data. This timeout is in order to prevent the user from entering wrong data due to a distraction, say, by a telephone call. This use case is captured by the UML2 sequence diagram shown below (in part).



3 Translating UML2 Interactions via Timed Automata into Timed CSP

We translate a (timed) UML2 interaction into a timed automaton [AD94] which can be seen as an observer of the message exchanges in a system reacting to these exchanges. The automaton registers the sending of a message as an event, and receiving a message as a separate event. Thus, the transmission time of a message can be considered. Asynchronous message passing is fully supported.

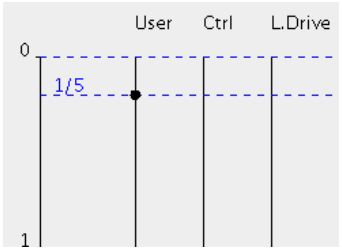
Our translation extends our previous work [KW07] now involving time. Abstractly, a timed UML2 basic interaction $((O,\preceq),\Gamma)$ consists of two components: on the one hand it comprises a partial order $(O,\preceq)$ where O is a finite set of events (observations) and $\preceq$ fixes which events have to *happen before* other events; on the other hand, a basic interaction comprises time constraints Γ on O. The translation of a timed basic interaction $((O,\preceq),\Gamma)$ into a timed automaton is performed by *unwinding* the partial order of events $(O,\preceq)$ in *phases* similar to the technique presented for live sequence charts [BDK+04], simultaneously taking into account the time constraints Γ.

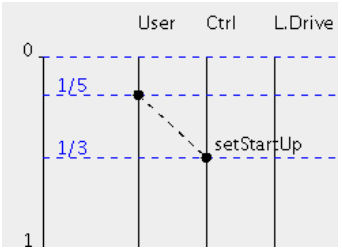
It is an interesting, however, not too involved exercise to systematically encode the resulting timed automaton in the process algebra Timed CSP, see e.g. [Sch00]. Here, we apply a construction which is far less involved than the one given by Ouaknine and Worrel [OW03].

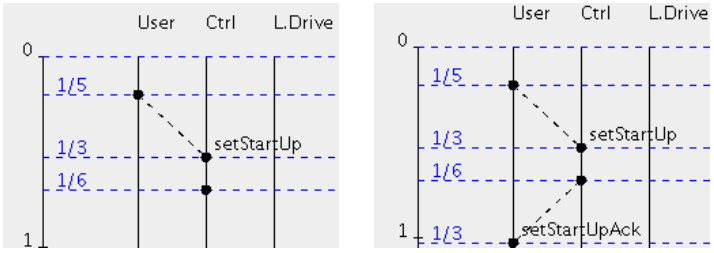
In general, our approach is prepared to handle all elements of Timed Sequence Diagrams with some restrictions to loops, negation, and dynamic object creation; currently, however, we focus on basic diagrams.

4 Simulation and Visualisation

The tool Timed CSP Simulator [FGM+12] allows to simulate Timed CSP in both ways described above: as an interactive exploration tool and via generation of randomized runs. However, as informative as such a simulation might be, it is problematic as it changes representation. Therefore, we translate the output of Timed CSP simulator back into a format close to UML2 sequence diagrams:







Here, the numbers in the left column are user or randomly chosen time delays. Via control buttons, the user can explore runs of a sequence diagram in a step-by-step fashion.

5 Conclusion

We have presented a prototypical, automated implementation of a tool chain from Timed UML2 Sequence Diagrams to an interactive visualisation tool simulating their different runs. The interactive visualisation allows to check in an explorative way whether the use cases have been modelled correctly. Our experiments concerned models of moderate size. A potential bottleneck in the tool chain is the translation into timed automata, which has an exponential worst case complexity in the number of observations [KW07]. However, "normal" diagrams do not pose a problem.

We base our current work on Timed-CSP, as – in the long run – this allows us also to consider time bounds in the Sequence Diagrams which are given as expressions, depending, e.g., on user input or the system state. Overall, this work is a first step towards developing a simulation and verification framework for timed UML2 – open for external tools dealing with Timed CSP, timed automata, and other formal methods.

Bibliography

[AD94] R. Alur, D. L. Dill. A Theory of Timed Automata. *Theo. Comp. Sci.* 126:183–235, 1994.

[BDK+04] M. Brill, W. Damm, J. Klose, B. Westphal, H. Wittke. Live Sequence Charts. In Ehrig et al. (eds.), *Integration of Software Specification Techniques for Applications in Engineering*. Lect. Notes Comp. Sci. 3147, pp. 374–399. Springer, 2004.

[FGM+12] M. Fontaine, A. Gimblett, F. Moller, H. N. Nguyen, M. Roggenbach. Timed CSP Simulator. In *Proc. Posters & Tool Demos Sess. iFM&ABZ'12*. 2012.

[KW07] A. Knapp, J. Wuttke. Model Checking of UML 2.0 Interactions. In Kühne (ed.), *Rev. Sel. Papers Wsh.s MoDELS'06*. Lect. Notes Comp. Sci. 4364, pp. 42–51. Springer, 2007.

[OW03] J. Ouaknine, J. Worrell. Timed CSP = Closed Timed ε-Automata. *Nord. J. Comput.* 10(2):99–133, 2003.

[Sch00] S. Schneider. *Concurrent and Real-time Systems*. Wiley, 2000.