VI. Summary

In this paper we have analysed technical aspects of code generation based UML model simulation. The main motivation for this solution is higher runtime performance, required by non-interactive model-level testing use cases. On the other hand, the same execution engine should also serve interactive model debugging sessions. The paper gives detailed descriptions of the techniques we found useful to solve this challenge.

The technical content of the paper is based on the design and implementation of a model simulator in industrial cooperation. Our experience with this tool confirms that the architecture and techniques presented in the paper are working well in practice. The incremental code generation, triggered by saving the model, is silently running in the background and does not have impact on the model editing process. We faced many technical difficulties to make this process work well: The life cycle of the different builders, integration of the incremental model query engine, interworking with the model editor and its resource handling processes are issues which all need to be solved correctly to make the user experience smooth.

So far, we have not found conceptual limitations of the code generation based approach. Most probably, it would be easier to show information about the runtime state of an interpreter than the internals of generated code running in a separate virtual machine. On the other hand, the sophisticated framework for debugging Java programs is a solid basis to build model debuggers on. The expected runtime performance gain over interpreters makes the approach worthwhile.

We experience that having a good model simulator is only one tiny piece of the infrastructure needed to make executable UML modelling work in an industrial setup: User friendly model editors, model validators, model compare-and-merge tools, version control and team work support, refactoring tools and model compilers are all needed to make a toolchain practical.

Acknowledgment

We express our gratitude to Ericsson for the financial support of this research.

References

[1] EMF-UML2 project. http://wiki.eclipse.org/MDT-UML2.
[2] Executable Translatable UML Open Source Editor. https://www.xtuml.org/.
[3] IBM. Rational Rhapsody family. www.ibm.com/software/products/en/ratirhapfami.
[4] Java Platform Debugger Architecture (JPDA). http://docs.oracle.com/javase/8/docs/technotes/guides/jpda/.
[5] Java Specification Requests 45: Debugging Support for Other Languages. https://jcp.org/en/jsr/detail?id=45.
[6] Javassist library. http://www.csg.ci.i.u-tokyo.ac.jp/~chiba/javassist/.
[7] Stefan Mijatov, Philip Langer, Tanja Mayerhofer, and Gerti Kappel. A Framework for Testing UML Activities Based on fUML. In *Proceedings of the 10th International Workshop on Model Driven Engineering, Verification and Validation (MoDeVVa) co-located with 16th International Conference on Model Driven Engineering Languages and Systems (MODELS 2013)*, pages 1–10, 2013.
[8] Moka. http://wiki.eclipse.org/Papyrus/UserGuide/ModelExecution.
[9] Object Management Group. Action Language for Foundational UML (ALF), standard, version 1.0.1. http://www.omg.org/spec/ALF/, 2013.
[10] Object Management Group. Semantics of a Foundational Subset for Executable UML Models (fUML), standard, version 1.1. http://www.omg.org/spec/FUML/1.1/, 2013.
[11] Object Management Group. Precise Semantics of UML Composite Structures (PSCS), standard in preparation, version 1.0 beta 1. http://www.omg.org/spec/PSCS/1.0/Beta1/, 2014.
[12] OneFact. BridgePoint xtUML tool. http://onefact.net/.
[13] Papyrus. http://wiki.eclipse.org/Papyrus.
[14] Nadege Pontisso and David Chemouil. Topcased combining formal methods with model-driven engineering. In *Automated Software Engineering, 2006. ASE'06. 21st IEEE/ACM International Conference on*, pages 359–360. IEEE, 2006.
[15] Sally Shlaer and Stephen J. Mellor. The Shlaer-Mellor method. *Project Technology white paper*, 1996.
[16] Technical Committee ISO/IEC JTC1, Information technology, in collaboration with the Object Management Group (OMG). Object Constraint Language (OCL). Standard, International Organization for Standardization, Geneva, Switzerland, April 2012.
[17] Topcased migrates to PolarSys. http://polarsys.org/topcased-migrates-polarsys.
[18] xUML-RT Model Executor. http://modelexecution.eltesoft.hu/.