Jan Jürjens, Kurt Schneider (Hrsg.): Software Engineering 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 77

Model-Driven Software Engineering in the openETCS Project: Project Experiences and Lessons Learned

Stefan Karg[1] Alexander Raschke[2] Matthias Tichy[2] Grischa Liebel[3]

Abstract: Model-driven software engineering in industrial practice has been the focus of different empirical studies and experience reports. Particularly, positive effects of model-driven software engineering have been reported in the domain of embedded and safety-critical systems. We report in this paper on the experiences of the openETCS European research project whose goal was to formalize the System Requirements Specification and to develop an open source reference implementation of the European Train Control System including open source modeling tools. Furthermore, we will discuss lessons learned, e.g., about using open source modeling toolchains in safety-critical contexts and about using the SCADE Suite for the development of the safety-critical parts.

Keywords: ETCS, safety-critical systems, modeling, experience report, open source

1 Introduction and Motivation

The goal of the openETCS project (`www.openetcs.org`) was to provide a formalization of the European Train Control System (ETCS) standard as well as a reference implementation for ETCS. The idea was to enable manufacturers to test their product derived from the specification document written in prose text (hence including ambiguities) against the reference implementation. This will improve the interoperability between different ETCS implementations of different manufacturers. The openETCS project used a model-driven approach. The idea was to build up an open source tool chain to avoid vendor lock-in not only for ETCS products, but also for the development process.

2 Method

In order to identify the lessons learned in the openETCS project, we used three sources of data. The main author was working actively in the openETCS project for more than one year and was one of the main contributors to the modeling work package. To collect a wider feedback on the openETCS project and the chosen development process and tools, we performed an online survey targeting the project members. It covered 60 questions (including demographic questions) about the effectiveness of the chosen MDE process and tools, which modules are suited to modeling and and which not, developer satisfaction and shortcomings of the MDE process and tools. Finally, to collect insights outside the project,

[1] Informatik Consulting Systems AG, Sonnenbergstr. 13, 70184 Stuttgart, stefan.karg@ics-ag.de
[2] Institute of Software Engineering, Ulm University, 89081 Ulm, firstname.lastname@uni-ulm.de
[3] Software Engineering Division, Chalmers | University of Gothenburg, Gothenburg, grischa@chalmers.se

we conducted a semi-structured interview with an experienced engineer from Siemens AG working in the software engineering of automatic train protection systems.

3 Lessons Learned

The original goal providing an open source tool chain was not achieved, because a tool evaluation came to the result, that no open source tool comes with a code generator certified according to EN50128, which is the relevant standard for developing safety-critical software for railway systems. Instead, the Ansys SCADE product family was used. The use of this tool and the use of modeling itself in the openETCS project has been seen mainly positive. However, we identified several challenges and open areas for improvement.

First, SCADE is an example of a highly complex model-driven engineering tool. It requires extensive knowledge and experience to use effectively. Here, one area of future work is to improve the usability of modeling tools and, particularly, identify user interaction patterns for good usability and user interaction anti patterns for bad usability.

Second, in realistic industrial situations extensive support for collaboration in modeling and versioning is required. Even though there exist some works on collaborative modeling and versioning todays industrial modeling tools are not suited well to support these scenarios.

Third, providing open source modeling tools and open source models and software for safety-critical systems are only a minor part of the required effort for a safety-critical system. Most effort goes into further aspects, like certification and developing safety-cases. This aspect is not well supported by the open source community, yet.

Fourth, while for widely used tools and technology, e.g., open source tools and libraries, answers for problems can typically be found on web communities like stackoverflow, this is not the case for tools like SCADE. Hence, we have to be better on building communities for users of modeling tools.

Finally, modeling tools need to be open for extendability. On the one hand, they need to offer APIs to be able to add functionality. On the other hand, the modeling tools need to make it easy to integrate custom code, e.g., C-code. While this might make formal verification impossible or at least difficult, this is simply required to effectively develop systems as we experienced many times in using model-driven software engineering for industrial embedded systems. Although, the study was conducted in the railway domain, most of the mentioned issues can be transfered to other domains, too. Problems with tool usability, support and collaborated editing features apply to all domains using modeling tools. More details are given in [Ka16].

References

[Ka16] Karg, Stefan; Raschke, Alexander; Tichy, Matthias; Liebel, Grischa: Model-driven Software Engineering in the openETCS Project: Project Experiences and Lessons Learned. MODELS ’16, ACM, New York, NY, USA, pp. 238–248, 2016.