S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 143

Using Language Workbenches and Domain-Specific Languages for Safety-critical Software Development

Markus Völter[1]

Abstract: In a 2018 article[2] in the journal on Software & Systems Modeling we[3] discussed the use of DSLs and language workbenches in the context of safety-critical software development. Language workbenches support the efficient creation, integration, and use of domain-specific languages. Typically, they execute models by code generation to programming language code. This can lead to increased productivity and higher quality. However, in safety-/mission-critical environments, generated code may not be considered trustworthy, because of the lack of trust in the generation mechanisms. This makes it harder to justify the use of language workbenches in such an environment. In the SOSYM paper, we demonstrate an approach to use such tools in critical environments. We argue that models created with domain-specific languages are easier to validate and that the additional risk resulting from the transformation to code can be mitigated by a suitably designed transformation and verification architecture. We validate the approach with an industrial case study from the healthcare domain. We also discuss the degree to which the approach is appropriate for critical software in space, automotive, and robotics systems.

Keywords: Domain-Specific Languages, Language Workbenches, Language Engineering, Safety-Critical Systems, Testing

1 Key Challenge

A language workbench (LWB) can be used to define domain-specific language (DSL) optimized for the application domain of a critical software component (CSC). Models are then used to describe one or more particular CSCs. The implementation of the CSC is automatically derived from the model. If the transformation to the executable code is correct, this leads to significant gains in productivity.

However, the approach can introduce additional failure modes because of errors in the LWB's generation framework or in the DSL-specific generators or interpreters.

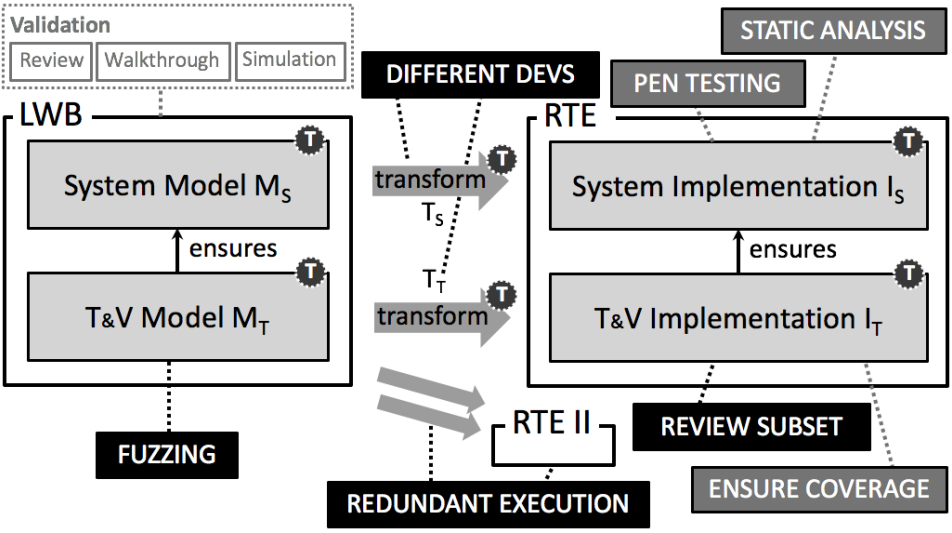
Correctness of the tool cannot be assumed because the LWBs and the DSLs cannot be argued to be qualified tools. So, how can a non-qualified LWB and custom-built DSLs be used in the development of critical systems?

[1] independent/itemis, Ötztaler Strasse 38, 70327 Stuttgart, Deutschland. voelter@acm.org

[2] https://link.springer.com/article/10.1007/s10270-018-0679-0

[3] coauthored by, Bernd Kolb, Klaus Birken , Federico Tomassetti, Patrick Alff, Laurent Wiart, Andreas Wortmann and Arne Nordmann

doi:10.18420/se2019-44



In effect, we are left with ensuring end-to-end correctness, from the model to the implementation; we treat the combination of model + DSL + generator + LWB as one "untrusted" black box. The challenge we address in this paper thus becomes:

How can a non-qualified LWB and custom-built DSLs be used in the development of critical systems, ensuring that the approach does not introduce faults into the CSC and continuing to exploit the benefits afforded by DSLs?

2 Solution

We express both the system and the tests or verification properties on model level and then translate both of them to the implementation and run them there. This way we *express and validate* the semantics on the convenient level of the model, but then *execute and verify* the semantics on the (ultimately relevant) implementation level.

We have performed a systematic analysis of the risks introduced by the DSL-based approach and use the techniques mentioned in the figure to address them.

3 Evaluation

We have evaluated the approach in a project in the healthcare industry, the paper contains a detailed case study. Voluntis developed a familiy of software medical devices. One successfully passed FDA pre-approval. Voluntis reported significant reductions in development and testing effort, as well as improvements in quality.