Model-Driven Design and Analysis of Component-Based Reactive Systems Abstract

Author
Bence Graics

Advisor
Vince Molnár
András Vörös

As the complexity of software systems increases, more and more responsibility falls onto system engineers who have to supervise the design and implementation of interacting system components. Extra effort has to be invested into the design and analysis of systems in the safety-critical domain. These processes can be supported well with design and analysis tools.

The gamma framework presented in this work is designed to support model-driven development of safety-critical systems. One of the greatest merits of the framework is that it is extensible with modeling tools (supporting design or analysis). In the current version Yakindu Statechart Tool is integrated into the framework to facilitate the modeling of state-based behavior. Additionally, gamma framework relies on the code generation functionalities of Yakindu.

The main functionality of the gamma framework is supporting the hierarchical composition of state-based models. This is achieved with a concise, easy-to-use textual compositional language. The behavior of the specified composite systems conforms to a well defined turn-based semantics. The compositional language is built upon our internally developed statechart language which functions as an intermediate language. This intermediate language enables the future extension of the framework with additional state-based formalisms.

To integrate a modeling tool into the gamma framework a model transformation has to be implemented which maps models to the intermediate statechart language. For this purpose we use the VIATRA model transformation framework. To support the modeling process, validation rules are defined for the intermediate statechart language to find design flaws as soon as possible.

Additionally, the gamma framework supports the derivation of source code form the hierarchical composite systems, thus creating composite system implementations. Here the simple statechart implementations are generated by Yakindu and the glue code implementing their interactions is produced by the code generator of the gamma framework. The generated code is Java and conforms to the operational semantics of the compositional language, naturally.

Additional model transformations are defined in the gamma framework that produce UPPAAL automata from the composite systems, thus supporting their formal verification. The generated network of UPPAAL automata follows the semantics of the compositional language and is conformant to the composite system implementation. In addition to UPPAAL automata, temporal logic expressions are generated which can be used by UPPAAL to generate test-cases verifying the reachability of each state in the composite system. What is more, the framework is capable of back-annotating the results of the formal analysis, so the states of the analysis models can be examined in the domain of the compositional language. Using the test-cases and the results of the back-annotation the conformance of the composite system implementation and the UPPAAL automata can be validated.