Abstract

As a result of the recent technological advancements in computation, programmable controllers are now used extensively even in critical domains such as automotive embedded systems. Moreover, in the era of "intelligent" devices, programs are not centralized anymore – for example, the embedded controller directly actuating the vehicle is in close relation with the electronic control unit, which is in turn communicating with services in the cloud. The complexity of such heterogeneous systems may be very high. Model-driven development is widely used to handle the complexity because it supports the developer in focusing on the logical aspects of the problem instead of the technical details.

In this work, we present a modeling tool to answer the following challenges inherent in the systems characterized above.

1. Component-based architecture: the targeted systems are typically composed of smaller components – therefore a suitable modeling language shall support hierarchical modeling.

2. Communication: components usually communicate by means of logical signals or messages – communication shall happen through well-defined ports and interfaces.

3. Distributed components: components often do not constitute a single program, but several pieces of software that run on different pieces of hardware – the resulting heterogeneous communication requires different compositional semantics.

4. Quality and correctness: often, these systems (or parts of them) perform critical tasks where correct operation is fundamental – therefore, their design must be sound and correct, which can be supported by validation and verification, while the quality of the implementation can be ensured with automatic code generation and testing.

In this work we propose the gamma framework, which is a modeling tool to build hierarchical, component-based, reactive systems. Elementary components can be defined in the built-in formal modeling language as well as in third-party tools integrated with gamma (e.g., Yakindu Statechart Tools). The framework supports three types of semantics for composition: asynchronous-reactive semantics for the proper abstraction of distributed communication, synchronous-reactive for components of a single program or for highly synchronous communication, and cascade for the logical decomposition of a single function. The modeling process is supported by live validation both on the component and system level. Model checkers (such as UPPAAL), integrated into the framework and hidden from the user, can be used to ensure the correctness of models. The implementation of the design can be automatically generated from the models, where the quality of the generated code is validated by a set of automatically generated tests.

The extensive functionality and the possibilities provided by the gamma framework are also demonstrated through a railway-themed case study.