Enabling the Interoperability of the Modelica DSL and Matlab Simulink towards the Development of Self-Adaptive Dynamic Systems

Nikolay Tcholtechev, Fraunhofer Institute for Open Communication Systems FOKUS, Berlin, Germany
Grit Dudeck, Fraunhofer Institute for Open Communication Systems FOKUS, Berlin, Germany
Michael Wagner, Fraunhofer Institute for Open Communication Systems FOKUS, Berlin, Germany
Christian Hein, Fraunhofer Institute for Open Communication Systems FOKUS, Berlin, Germany
Arun Prakash, Fraunhofer Institute for Open Communication Systems FOKUS, Berlin, Germany
Tom Ritter, Fraunhofer Institute for Open Communication Systems FOKUS, Berlin, Germany

ABSTRACT

Domain Specific Languages (DSL) are an important concept that is used in industry, in order to enable the fast and cost-efficient design of specific functions/components, and/or to target particular aspects of the systems’ development and operation. In the current article, the authors describe their experiences on the integration of the Modelica DSL into a platform that enables the integration and interoperability of model-based tools across the various phases of the system development process. Furthermore, it is illustrated how Matlab Simulink can be used in parallel in the course of the same system design undertaking. Thereby, the authors present their approach and compare different tools which were used, in order to efficiently complete the integration, and finally exemplify the outcome on a case study related to a self-adaptive dynamic system from the automotive domain.

KEYWORDS
Domain Specific Languages, DSL, MDE, Model Driven Engineering, ModelBus, Modelica, Modeling, Simulink, Tool Interoperability

1. INTRODUCTION

Domain specific languages (DSL) are an important concept that is used in industry, in order to enable the fast and cost-efficient design of specific functions/components, and/or to target particular aspects of the systems’ development and (runtime) operation – for instance configuration management or network routing policies. Thereby, the abstractions which are provided by a general-purpose modeling language, such as UML, are left aside in order to efficiently target the specific aspects under consideration. In that line of thoughts, the Modelica DSL constitutes an object-oriented and declarative - in terms of listing down equations and components - language for the specification of various types of complex systems, including mechanical, electrical, thermal, and other types of systems. The Modelica standard has developed over the past years and a number of tools are available which enable the modeling, specification, simulation, verification, and iterative improvement of the system model in question.

DOI: 10.4018/IJSDA.2018100104

Copyright © 2018, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
Therefore, Modelica emerges as an important language that is to be considered in the overall process of the development of complex dynamic systems in general. This overall process includes aspects ranging from requirements analysis, architectural specification and simulation, to implementation and testing of the system under development.

In that context, the current paper describes our experiences on the integration of the Modelica DSL into a platform that enables the integration and interoperability of model-based tools across the various phases of the system development process. Furthermore, the current paper illustrates how Matlab Simulink – an established framework for dynamic systems modeling and simulation - can be used in parallel in the course of the same system design undertaking. The importance as research work is constituted by (1) demonstrating that the aforementioned interoperability/integration between two major languages for modeling dynamic systems - i.e. Modelica and Matlab Simulink - is indeed possible, by (2) identifying a set of freely available tools and “how-to’s” regarding the realization of this challenging task, and finally by (3) demonstrating the benefit of the overall approach based on a case study relating to the development of a self-adaptive dynamic system from the automotive domain.

The platform for Model-based Tool interoperability and integrations is given by the ModelBus (ModelBus, 2018) (Armengaud et al., 2011) (Hein et al., 2009) (Aldazabal et al., 2013), which is a backend type of framework providing means for the integration of various model-based tools across the system design phases. Thereby, we describe the ModelBus extensions which are required in order to integrate Modelica into an established system development process running on top of ModelBus. In addition, the various tools are presented and technologies compared, which are used to obtain the different artefacts (e.g. metamodels) required for the Modelica integration into ModelBus. To fully integrate Modelica into an established system development process, key Model-2-Model (M2M) transformations are conceptualized and implemented on top of the ModelBus, in order to enable the smooth interoperability between Modelica tools and other tools across the system design phases. All the concepts are exemplified on a case study from the automotive domain that shows how Modelica can be used within the process of designing a car cruise controller, which constitutes a critical self-regulating system within a car.

With respect to Matlab Simulink, we build on an available bidirectional transformation between Simulink and SysML, which eventually enables the usage of Modelica and Matlab Simulink in parallel, for the purpose of systems design and simulation. The resulting transformation(s) are briefly described in the relevant sections.

The rest of this paper is organized as follows: Section 2 elaborates on related work. Section 3 describes the ModelBus platform in brief. Section 4 introduces the reader to the Modelica DSL. The following section 5 describes the ModelBus extensions which are required in order to integrate Modelica into ModelBus. In section 6, we share our experiences obtained in the course of establishing the Modelica metamodel required for the ModelBus integration. Thereby, we compare the main tools which were tried out in order to effectively create a Modelica metamodel with minimal deviation from the standard. Section 7 describes the principles behind the implementation of M2M transformations required for the integration of Modelica and corresponding tools into an established system development process on top of ModelBus. The following section 8 discusses on the integration of Matlab Simulink and elaborates on technical details regarding the required transformation. Section 9 shows how the concepts and resulting implementations are applied to a self-adaptive case study from the automotive domain. Finally, the last section summarizes our contribution, draws conclusions, and outlines future research directions.

2. RELATED WORK

The topic of model-based tool integration and interoperability has drawn much attention in the past years. Tools such as DOORS (Elizabeth et al., 2011; DOORS, 2018), Reqtify (Reqtify, 2018) and Rhapsody gateway (IBM Corporation, 2010) aim at setting relations and enabling traceability between
A Knowledge-Based System for Sharing and Reusing Tacit Knowledge in Robotic Manufacturing
Lei Wang, Yajie Tian, Tetsuo Sawaragi and Yukio Horiguchi (2012). Systems Approaches to Knowledge Management, Transfer, and Resource Development (pp. 308-326).
www.igi-global.com/chapter/knowledge-based-system-sharing-reusing/68227?camid=4v1a

Designing for Changing Work and Business Practices
www.igi-global.com/chapter/designing-changing-work-business-practices/4218?camid=4v1a