Available online at www.sciencedirect.com

ScienceDirect

IFAC-PapersOnLine 48-24 (2015) 111–116

Correct and Cost-efficient Development of Control Systems Based on Model Integration

Habibe M. Eminova, Iskra D. Antonova, Idilia A. Batchkova

Department of Industrial Automation, University of Chemical Technology and Metallurgy, Sofia, Bulgaria (Tel: +35928163326; e-mail: {h_eminova, iskra.antonova, idilia}@uctm.edu)

Abstract: The increasing demands on functionality and quality of the software in control systems are in contradiction with the increasingly shorter deadlines for its development and insufficient budget. The majority of software errors occurs in the early phases of its development, but is detected in later phases of testing. Later identified errors lead to extremely high costs for their removal. To overcome these drawbacks, new approaches for development of real-time software are necessary. The main objective of this paper is to present a framework for model-driven development of control systems based on UML and reinforced with additional opportunities for an early analysis and verification of the models. For this purpose an approach for integrating UML state machine and timed automata models has been developed and used. The integration is achieved through model transformation, based on meta-models and the transformation language ATL, plug-in in Eclipse platform. The proposed approach is illustrated with a simple example for discrete-event control of air compressor system. Finally, some conclusions are made.

Keywords**:** Control systems, State machine, Timed Automata, model transformation, formal verification

1. INTRODUCTION

In contemporary flexible and customer-oriented automated manufacturing, the development of software for control systems is a significant factor that requires considerable amount of time and money. Moreover, the quick adaptation of control algorithms to new manufacturing scenarios is increasingly necessary. The growing complexity of advanced software and rising customer requirements concerning dependability and functionality requires the use of new approaches to the development of real time software.

Development and implementation of control systems include variety of diverse tasks such as model design, development of simulation models, analysis, verification and validation of models as well code creation using specific formalisms and specialized software tools. These formalisms are highly divergent, i.e. for their development different approaches in respect to their syntax, semantics, computational models and performance are used. Lack of interoperability between different phases of software development and the combined use of different modelling tools affects the quality and efficiency of the software being developed.

In order to improve the overall approach for development of control systems, it is desirable to interconnect different modelling paradigms, so as to enable the sharing of data and model fragments between different formalisms and tools that support them. There are two basic approaches for solving this problem: the first one is the creation and use of common standardized language and the second one is based on the idea of using meta-models. Unifying all aspects of heterogeneous modeling paradigms in common standardized language means, however, a number of compromises, which will significantly reduce the usability of the language. Therefore, the authors give preference to the second approach, which may be applied for definition of modelling languages, domain modelling as well model transformations. The interest in model transformation, which is the core of the submitted work for model integration, is mainly related to the following applications in the domain of software and control engineering:

- Transformation to executable specifications;
- Transformation to formal models for analysis, verification and validation;
- Transformation of platform-independent models into platform-specific models.

One of the approaches that is able to successfully solve the task of model integration based on meta models is the so-called Model Driven Engineering (MDE) (Kent, 2002) or its counterpart in the field of architectures - Model Driven Architecture (MDA) (OMG-MDA, 2003), (Frankel, 2003). The Unified Modelling Language (UML) (OMG-UML, 2010) as a general purpose modelling language and an open standard supports the MDE and MDA. UML gives uniform notations and a meta-model for object-oriented modelling and software design providing an integrated modelling framework, covering structural, functional and behaviour descriptions. The approaches of MDE and MDA consist in transformation of different platform independent models towards platform specific applications. The same idea may be applied to the task of transformation to formal models

10.1016/j.ifacol.2015.12.066

for analysis, verification and validation or transformation to executable specifications.

UML State Machine diagrams (SM) play an essential role in the software process model for development of control systems and are used in the early phases of their development for modelling the behaviour of their components.

The main aim of the proposed study is to create an approach for model transformation of UML 2.0 state diagram into timed automata in order to support the analysis, verification and validation of control systems. The transformation is based on the definition and use of meta-models and set of transformation rules according ATLAS Transformation Language (ATL) (Allilaire *et al.*, 2006). As a powerful tool for analysis, verification and validation of generated Timed Automata (TA) model, UPPAAL is used.

The paper is organized as follow. After the introduction, in the next session, related works in the fields of MDD and use of formal methods in software models are discussed. In part 3 of the paper the proposed approach for model integration is described. Part 4 presents a simple case study concerning an application of the suggested approach. Finally some conclusions are made and some aspects of the future research are discussed.

2. RELATED WORK

2.1 MDD and model transformation languages

According to the MDA approach, each model is based on a meta-model that describes its elements and relations. OMG provides a special standard QVT (Query View Transformation) for model transformation (OMG-QVT, 2001). The basic idea of transformation is its presentation as a model, which can be transformed into models on the different levels of abstraction. The rules for transformation in QVT are defined using OCL, and the transformation includes transformations of platform independent to platform-dependent models, the definition of views and synchronization between different models. A major shortcoming of this standard is that transformation includes only models, whose meta-meta-model is the Meta Object Facility (MOF) (OMG-MOF, 2001). In order to overcome this drawback the proposed work uses the ATLAS Transformation Language (ATL) (Allilaire *et al.*, 2006). ATL is introduced by the ATLAS INRIA & LINA to support the implementation on the MOF/QVT standard. Its main purpose is to provide ways to create a kind of models from other source models in the Eclipse platform. Transformation in ATL is based on a set of rules, which describe how the elements of the source model are initialized and navigated to create the elements of the target model. Beside rules ATL also provide a set of helpers which can be defined and called from different points of an ATL transformation module. The language offers two styles of writing to transformation rules - declarative and imperative. Rules providing declarative style also called matched rules are expressed in directly mapping between source and target element. The rules providing imperative programming facilities are called rule. In ATL, helpers are viewed as Java methods in declarative style.

2.2 Software models based on formal methods

Formal specification, modelling and verification represent different aspects of the designed software system and all they are included as phases in the contemporary software models. This has a significant impact on the quality and efficiency of the software being developed. On the other hand, as noted above, software models are increasingly based on the use of MDD and UML. Striving formalization of UML for now is insufficient to provide adequate, reliable and efficient means of analysis, verification and validation of the models used. But the fact is that there are many successful methods and developments based on sustainable formalisms that allow analysis, verification and validation of different types of models such as different types of machines, Petri nets, abstract machines, transition systems, etc.

The development of software for control applications may use different software models as represented on Fig.1. The middle chain in the picture proposes the application of formal approaches and is based on the following main processes:

- Informal specification describing the system requirements;
- Informal verification of system requirements;
- Formalization process used to achieve the formal specification (model) by using different techniques as for example timed, finite or hybrid automata, different kinds of Petri nets etc.
- Formal verification process to early proving the satisfaction of system requirements by using different techniques such as Binary Description Diagram (BDD), propositional satisfiability (SAT), reachability graphs and many others.
- Implementation, which is defined as a process of automated code generation by using the designed and verified formal model.
- Processes of finally verification and validation based on already realized software system.

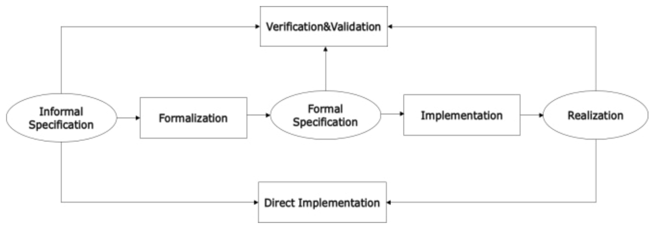

Fig. 1. Control system development life cycles (Frey, 2002)

UML 2.0 SMs are important, well-structured, useful and visual modeling instruments for modeling of behaviour and execution semantics of systems. The state machine diagram is a suitable UML diagram for modelling of system's component behaviour. The overall system behaviour can be