IMPLEMENTING DEVS MODELS WITH CADMIUM SIMULATOR

Cristina Ruiz-Martin
Dept. of Systems and Computer Engineering
Carleton University
Ottawa, Canada
cristinaruizmartin@sce.carleton.ca

Gabriel Wainer
Dept. of Systems and Computer Engineering
Carleton University
Ottawa, Canada
gwainer@sce.carleton.ca

ABSTRACT

Discrete Event System Specification (DEVS) is a mathematical formalism to model and simulate discrete-event dynamic systems. Using DEVS for modeling and simulation has numerous advantages, which include a rigorous formal definition of models, a well-defined mechanism for modular composition, and separation of concerns between the model definition and the simulation of the model, among others. In this tutorial, we will present Cadmium, a new DEVS simulator based on C++17. We will discuss the tool's Application Programming Interface and we will present the model of the Alternating Bit Protocol as an example to explain how to implement DEVS models in Cadmium.

Keywords: DEVS, Modeling & Simulation, Simulation tools.

1 INTRODUCTION

Discrete EVent System specification (DEVS) (Zeigler et al. 2000) is a hierarchical and modular formalism for modeling Discrete Events Systems (DES). The hierarchical and modular structure of DEVS allows defining multiple sub-models that are coupled together.

In DEVS, there are two kinds of models: atomic, which defines the behavior of the system, and coupled, which describes the structure of the system. The transition functions implement the behavior and change the state of the system. In addition, when the system state changes, the atomic model decides how long the system stays in the new state until an internal transition is triggered. Before every internal transition occurs, the output function is called to send an output through an output port.

The use of DEVS provides several advantages in the field of modeling and simulation. It is a methodology to develop hierarchical models in a modular way. This modularity allows model reuse and thus, reducing development time and testing. The model definition, implementation, and simulation are separated. The same model can be implemented on different platforms easing reliability and correctness.

In this tutorial, we present one of these DEVS simulation platforms: *Cadmium*. *Cadmium* is a new DEVS simulator based on C++17 and easy to include in different projects.

2 CADMIUM

Cadmium is a tool for Discrete-Event modeling and simulation, based on the DEVS formalism. Cadmium is a cross-platform header-only library implemented in C++. A complete user's guide to Cadmium can be found in (Ruiz-Martin and Wainer, 2019).

During the 1.5h tutorial, we will focus on explaining the tool's Application Programming Interface and how to implement a model following the example provided in the user's guide. Here, we just summarize the main advantages and features of Cadmium.

One of the main advantages of Cadmium is that we can use different data types for messages in different ports. Therefore, it eliminates a restriction of using a single type of data for all the messages that exits in many DEVS simulators developed so far.

Cadmium also implements other advanced features that help with model verification and reduces the probability of running a simulation with a bug. It implements model checks for both atomic and coupled models. Some checks are in compilation time and others in execution time, but all of them before a simulation starts.

For atomic models it implements all the following checks:

- Model methods: we check whether all the methods of the atomic model (internal, external, time advance, output and confluence) are defined, and they have the correct parameters and return types.
- Ports: we check that the model port types are correct.
- Valid model state type: we check if the model class has an attribute called *state* of type model name::state type. This assures that the model is defined and allows the simulator to verify that the state is changed only in the transition functions (i.e. internal, external and confluence).

For coupled models, Cadmium checks include:

- Link types consistency: the model should not connect two ports with different message types.
- Connected ports: the model should not connect invalid ports based on the EIC, EOC and IC structure.
- Valid coupled and atomic submodels: the simulator checks recursively that the submodels are valid (i.e. the atomic model checks are performed).

REFERENCES

Ruiz-Martin, C., and G. Wainer. 2019. *Cadmium. A tool for DEVS Modeling and Simulation. User's Guide*. Available online: http://www.sce.carleton.ca/courses/sysc-5104/lib/exe/fetch.php?media=cadmium.pdf. Last access 02/01/2020.

Zeigler, B.P., H. Praehofer, and T. G. Kim, 2000. *Theory of modeling and simulation*. Academic Press.

AUTHOR BIOGRAPHIES

CRISTINA RUIZ-MARTIN is a Postdoctoral Fellow at the Department of Systems and Computer Engineering at Carleton University. Her email address is cristinaruizmartin@sce.carleton.ca.

GABRIEL WAINER is a Full Professor at the Department of Systems and Computer Engineering at Carleton University. He is a Fellow of SCS. His email address is gwainer@sce.carleton.ca