INTRODUCTION TO CLASSIC DEVS

Yentl Van Tendeloo
University of Antwerp

Antwerp, Belgium
yentl.vantendeloo@uantwerp.be

Hans Vangheluwe
University of Antwerp
Flanders Make, McGill University
Antwerp, Belgium
hans.vangheluwe@uantwerp.be

ABSTRACT

DEVS is a popular formalism to model system behaviour using a "discrete-event" abstraction. With this abstraction, a finite number of pertinents "events" can occur during any bounded time interval. In reaction to events, the model's state variable values change instantaneously. The state remains unaltered between event occurrences. The main advantages of DEVS are its rigorous and precise specification, as well as its support for modular, hierarchical construction of models. DEVS frequently serves as a simulation "assembly language" to which models in other formalisms are translated, either giving meaning to new (domain-specific) languages, or reproducing semantics of existing languages. Models in different formalisms can hence be meaningfully combined by mapping them onto common denominator DEVS.

This tutorial introduces the basics of the Classic DEVS formalism. The formalism is explained in a bottom-up fashion, starting from simple autonomous Atomic (i.e., non-hierarchical) DEVS models, over Coupled DEVS models, up to the specification and running of simulation experiments. Each individual aspect of the DEVS formalism is introduced by means of a minimal running example: a simple traffic light. While the focus is on the practical use of DEVS modelling and simulation, the necessary theoretical foundations are interleaved, albeit at a high level of abstraction. More advanced topics, such as closure under coupling and flattening, modularity, and the abstract simulator, are briefly mentioned. The tutorial concludes with an overview of problems that can be solved by Classic DEVS modelling and simulation. Examples are presented using PythonPDEVS, though the foundations and techniques apply equally well to other DEVS simulation tools. All tools and examples can be found online at http://msdl.cs.mcgill.ca/projects/DEVS/PythonPDEVS.

This tutorial presents the basics of the Classic DEVS formalism. Classic DEVS is a discrete-event formalism

with a long history, which started in 1976. The tutorial is primarily based on the influential book "Theory of Modelling and Simulation" (Zeigler, Praehofer, and Kim 2000). In constrast with these publications, the tutorial gives a bottom-up, example-driven explanation of the various aspects of Classic DEVS. While the theoretical foundations are touched upon, they are interleaved with a concrete application – a traffic light– of the theory. Starting from a simple model of an autonomous traffic light, the model is extended to describe increasingly more complex behaviour of the traffic light, eventually using all Classic DEVS language constructs. The tutorial is example-driven and uses PythonPDEVS (Van Tendeloo and Vangheluwe 2014), a simple Classic DEVS simulator written in the high-level, interpreted language Python.

DEVS is a discrete-event formalism, so the tutorial starts with a brief description of this abstraction. In discrete-event models, a finite number of pertinents "events" may occur during any bounded time interval. In reaction to these events, a model's state variable values change instantaneously. The state remains

unaltered between event occurrences. The tutorial shows example input/state/output traces, and how they relate to PythonPDEVS models.

The basic building blocks of any DEVS model are the atomic DEVS models. These are atomic in the sense that they focus exclusively on defining behaviour (as opposed to structure). The tutorial starts with a simple model of an autonomous traffic light. This model has the three basic elements of a Classic DEVS model: state, time advance, and an internal transition function. While the behaviour of the traffic light is now modelled, there is not yet any way to actually see what is happening: states are private, for modularity reasons. The models is hence extended with two new elements: the output set and the output function. Traffic lights do however not always work completely autonomous, but rather react to external stimuli. Therefore, the model is again extended to deal with external events: the input set and the external transition function.

With behaviour of Atomic models specified, their interaction structure can now be defined, in Coupled DEVS models. DEVS models can send/receive events through output/input ports. By connecting ports, models can influence each other's behaviour. In addition to internal couplings between Atomic models, a Coupled model also specifies the coupling of its own ports to those of its constituent models. The interface of Atomic and Coupled models is identical which supports modular hierarchical composition. To enable correct reuse, a translation function takes the output of one model and translates it to match the expected type of input of a model it is connected to. This is particularly helpful when reusing library models. Finally, when multiple atomic models want to perform their internal transition, a choice has to be made which one goes first. This is done by invoking a select function, which returns the model that transitions first.

One of the main advantages of DEVS is its formal foundation, covering syntax and semantics. The tutorial presents the semantics of a DEVS model in two ways: by flattening of coupled models (denotational semantics) and in the form of an abstract simulator algorithm (operational semantics). The tutorial restricts itself to an informal description of both semantics as that is sufficient for most common uses of the formalism.

REFERENCES

Van Tendeloo, Y., and H. Vangheluwe. 2014. "The Modular Architecture of the Python(P)DEVS Simulation Kernel". In *Proceedings of the 2014 Spring Simulation Multiconference*, 387–392.

Zeigler, B., Praehofer, H. and Kim, T. 2000. Theory of Modeling and Simulation. 2nd ed. Academic Press.

AUTHOR BIOGRAPHIES

YENTL VAN TENDELOO is a PhD student in the department of Mathematics and Computer Science at the University of Antwerp (Belgium). In his Master's thesis, he worked on MDSL's PythonPDEVS simulator, a simulator for Classic DEVS, Parallel DEVS, and Dynamic Structure DEVS, grafted on the Python programming language. The topic of his PhD is the conceptualization, development, and distributed realization of a new (meta-)modelling framework and model management system called the Modelverse.

HANS VANGHELUWE is a Professor in the department of Mathematics and Computer Science at the University of Antwerp (Belgium) and an Adjunct Professor in the School of Computer Science at McGill University (Canada). He heads the Modelling, Simulation and Design (MSDL) research lab. He has a longstanding interest in the DEVS formalism and is a contributor to the DEVS community of fundamental and technical research results.