Simulation Modelling Practice and Theory 82 (2018) 116–131

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Activity-based DEVS modeling

Abdurrahman Alshareef[a], Hessam S. Sarjoughian[a,*], Bahram Zarrin[a,b]

[a] Arizona Center for Integrative Modeling & Simulation, School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, 699 S. Mill Avenue, Tempe, AZ, 85281, United States
[b] DTU Compute, Technical University of Denmark, Kgs Lyngby 2800, Denmark

ARTICLE INFO

Article history:
Received 13 May 2017
Revised 16 November 2017
Accepted 12 December 2017

Keywords:
Activity modeling
Behavioral modeling
GMF
Model Driven Development
DEVS
UML

ABSTRACT

Use of model-driven approaches has been increasing to significantly benefit the process of building complex systems. Recently, an approach for specifying model behavior using UML activities has been devised to support the creation of DEVS models in a disciplined manner based on the model driven architecture and the UML concepts. In this paper, we further this work by grounding Activity-based DEVS modeling and developing a fully-fledged modeling engine to demonstrate applicability. We also detail the relevant aspects of the created metamodel in terms of modeling and simulation. A significant number of the artifacts of the UML 2.5 activities and actions, from the vantage point of DEVS behavioral modeling, is covered in details. Their semantics are discussed to the extent of time-accurate requirements for simulation. We characterize them in correspondence with the specification of the atomic model behavior. We demonstrate the approach with simple, yet expressive DEVS models.

1. Introduction

Constructing simulation models, despite from being quite costly and complex, remains indispensable and highly beneficial, especially for systems that do not lend themselves to analytical methods. System dynamics need to be determined in enough details to sufficiently address its different aspects under study in order to ultimately attain the potential benefit. The models have to be then realized in certain computational and physical environments in order to enable the simulation and experimentation thereafter. As system complexity grows, so does the importance of behavioral modeling. There are existing concepts and techniques where the structure modeling can be handled systematically to account for further system complexity and growth. However, these techniques fall short with respect to behavioral modeling. Increasingly behavioral models are becoming large and therefore difficult to understand, formulate, and maintain using conceptual (informal) and mathematical modeling as well as their implementation in programming languages and evaluations.

The Discrete EVent System specification (DEVS) formalism [1] can effectively serve as a basis for simulation-based design and formulation of modular, component-based system models. The parallel DEVS formalism, based on time, input, output, states, and state transition, is widely supported by simulators, DEVS-Suite [2] for example, that have been implemented in different computing environments. Simulators need to serve different needs through complementary advanced capabilities including action-based behavior specification. The input for simulators is mainly models although the simulators significantly differ in the form by which the models have to be formulated, simulated, and evaluated. This has led to the rise of utilizing

* Corresponding author.
E-mail addresses: alshareef@asu.edu (A. Alshareef), sarjoughian@asu.edu, hessam.sarjoughian@asu.edu (H.S. Sarjoughian), baza@dtu.dk (B. Zarrin).

https://doi.org/10.1016/j.simpat.2017.12.009
1569-190X/

the so-called Model-Driven Engineering in simulation especially with respect to the focus on creating platform-independent models. Models of this nature are less inclined to carry details specific to the execution environments and therefore can be used and maintained for a wider set of M&S platforms, a key benefit of Model-Driven Architecture (MDA) [3]. The definitions in this approach have been proven to be consistent with the theory of modeling and simulation in multiple occasions [4–8]. In fact, it is desirable to have models of this nature in order to allow modelers to be more focused on the problem and solution specifications in more neutral terms with respect to specific details of simulation frameworks. Behavior specification is not as simple as structure specification especially when system dynamics require understanding and formulation beyond conditional state changes and event handling. This fact is accounted for the recent approaches that adopt some of the other languages and formalisms (e.g., UML state machines) with capabilities that can afford specifying complex behaviors.

While some behavioral languages are adopted for the specification of DEVS atomic models, they significantly differ in their suitability, complexity, and provided capabilities. In this work, we attempt to dig deep into the activity modeling as a major modeling approach for the DEVS atomic model and by extension coupled model behavioral specifications. The approach is gaining more attention given its promising prospects for enhancing system modeling in multiple domains. Activity metamodel has also undergone major advances in the recent decade especially the release of the UML 2.0 [9] and the foundational subset of UML (fUML) [10]. The idea essentially is to adopt the UML activities for the behavioral specification of the DEVS atomic model according to the state of the art standards. Activities provide some unique capabilities with respect to other behavioral diagrams. We want to leverage them in a way that shortens the distance between the concrete models and their mathematical abstractions. The handling of actions in the activities gives a premise to overcome some of the behavioral modeling difficulties in general and the ones encountered in the other behavioral approaches (e.g., finite-state machines). A richer specification can be achieved when modelers consider a variety of behavioral specifications that better serve their needs.

We will discuss some necessary background in this subject. We will also compare and contrast our contribution with some of the existing approaches toward meeting the need for expressive behavioral modeling. We elaborate on the selected approach based on previous work [11] in conjunction with further discussion and alignment with the DEVS formalism. A modeling engine is created to manifest that at the implementation level alongside with processor with queue model as an exemplar.

2. Background

There exist formalisms, modeling languages, and frameworks to develop behavioral models. Our work is centered on the rigorous specification of DEVS as an abstract mathematical formalism accompanied with a framework supported with modeling languages and run-time execution. In the following sub-sections, we describe basic background details for understanding and developing behavioral models.

2.1. Parallel DEVS atomic model

The set-theoretic specification of the atomic model is an abstract representation of a standalone component of a system [1]. The formal specification can be defined independent of any language, and more generally simulation platforms. From a software standpoint, we need to have the specifications to be formulated in terms of a modeling and software programming languages. There are many DEVS simulators that can accept the specification of a model following a target simulator's programming language syntax, semantics, and specialized constructs such as model initialization. Examples of these tools are DEVS-Suite [2] and CoSMoS (Component-based System Modeling and Simulation) [12] where the programming language is Java. Other simulators use different languages as an input such as CD++ [13] and PowerDEVS [14] where the programming language is C++. In [15], the work provides a specific language based on the formal specification definition language with set of rules to translate it into simulatable models targeting simulators like DEVS-Suite and PowerDEVS. As defined in [1], the basic formalism of parallel DEVS model is an algebraic structure – atomic model = $\langle X, Y, S, \delta_{ext}, \delta_{int}, \delta_{con}, \lambda, ta \rangle$. X is the set of input events. S is state representing the tuple of sequential states. The state must have at least two independent variables. One is called *sigma* (σ), the time duration allocated to the current state of the model. The other variable, called *phase*, represents a set of state values that change and be tracked. Y is the set of output events. δ_{int} and δ_{ext} are the internal and external transition functions, respectively. The model receives a bag of inputs meaning that the elements of the bag may have multiple occurrences and have no ordering. The receiving model accounts for this possibility in order to perform a proper handling of the inputs. δ_{con} is the confluent transition function which can be specified to handle the collision between external and internal events. λ is the output function which transforms S into Y at arbitrary time instances. ta is the time advance function which maps the internal state into a positive real number using elapsed time since last state transition (i.e., it computes σ which can range from zero to infinity, inclusive). Any domain specific definition of the aforementioned functions must satisfy their corresponding abstract definitions as provided in the modeling formalism. Together the elements of the DEVS specification allow modeler to flexibly define operations and controls for system structure and behavior.