Chapter 2.13
DEVS-Based Simulation Interoperability

Thomas Wutzler
Max Planck Institute for Biogeochemistry, Germany

Hessam Sarjoughian
Arizona Center for Integrative Modeling and Simulation, USA

ABSTRACT

This chapter introduces the usage of DEVS for the purpose of implementing interoperability across heterogeneous simulation models. It shows that the DEVS framework provides a simple, yet effective conceptual basis for handling simulation interoperability. It discusses the various useful properties of the DEVS framework, describes the Shared Abstract Model (SAM) approach for interoperating simulation models, and compares it to other approaches. The DEVS approach enables formal model specification with component models implemented in multiple programming languages. The simplicity of the integration of component models designed in the DEVS, DTSS, and DESS simulation formalisms and implemented in the programming languages Java and C++ is demonstrated by a basic educational example and by a real world forest carbon accounting model. The authors hope, that readers will appreciate the combination of generalness and simplicity and that readers will consider using the DEVS approach for simulation interoperability in their own projects.

INTRODUCTION

Interoperability among simulators continues to be of key interest within the simulation community. A chief reason for this interest is the existence of heterogeneous legacy simulations which are developed using a variety of programming practices and software engineering approaches. For many studied problems there exist already simulations models. Much work has been dedicated to develop the models, estimate parameters, verify the simulations and validate the model assumptions by comparing model results to observations. Hence, it is desirable to reuse such existing models

DOI: 10.4018/978-1-61350-456-7.ch2.13

for additional tasks and to integrate parts of several existing models into a common framework. However, this integration is hampered by the fact that the existing models have been developed in different programming languages, with different software and different software engineering approaches. There exist already several approaches to implement a joint execution and some of them are summarized in the background section. In this book chapter we suggest an additional approach to realize interoperability among disparate simulators that utilizes the Discrete Event System Specification (DEVS). We argue that this approach yields several benefits. One of the most important benefits is the reduced complexity for adapting heterogeneous legacy models.

The interoperability issue will be exemplified with a typical problem which we encountered in our work. The task was to construct a forest ecosystem carbon tracking model. With such a model the exchange of carbon between the forest ecosystem and the atmosphere can be studied. One can then implement and test several scenarios of forest management and simulate how these influence the exchange of carbon dioxide with the atmosphere to explore how these management strategies influence atmosphere carbon concentrations and global warming. There already existed a model for the stand growth, i.e. the growth of trees within a plot, which was based on extensive measurement of tree properties within the study region (Appendix A). It accounted for competition among different trees and incorporated knowledge of many species. Hence, the source code was quite extensive. There already existed also a model for the soil carbon balance that have been used and validated by many research groups (Appendix B). Only the management strategies had to be implemented as a new component model.

However, the stand growth model was specified as discrete-time model (DTSS) and implemented in JAVA and the soil carbon balance model was specified as continuous (differential equations) model (DESS) and implemented in C++ (see Figure 1). So how could we couple the heterogeneous models and perform a joint and integrated simulation?

The purpose of this chapter is to discuss the benefits of DEVS concept compared to several commonly applied alternatives of tackling the problem of simulation interoperability. To this end we introduce the Shared Abstract Model

Figure 1. Motivation example: Simulating a coupled forest ecosystem carbon budget model which is composed of component models that have been specified in simulation formalisms and programming languages