DVML: DEVS-Based Visual Modeling Language for Hybrid Systems*

Hae Young Lee, Ingeol Chun, and Won-Tae Kim

CPS Research Team, ETRI
Daejeon 305-700 Republic of Korea
haelee@ieee.org

Abstract. This paper proposes a visual modeling language for a large-scale hybrid system based on discrete event system specification formalism. Within the proposed language, basic model diagrams define the hybrid behavior of systems, while the structure can be represented through coupled model diagrams. The language can visually define large-scale hybrid systems without a sacrifice of formal semantics, and does not require the users to have programming skills. A prototype of the modeling and simulation environment based on an extended version of the language has been implemented.

Keywords: Visual modeling language, discrete event system specification (DEVS), DEV&DESS, hybrid system modeling, cyber-physical systems.

1 Introduction

Discrete event system specification (DEVS) formalism [1], which is a theoretically well-grounded means of expressing modular discrete event simulation models, has gained increasing popularity in recent years [2]. Various DEVS-based modeling and simulation (M&S) environments [3,4] have been developed to tackle complex problems through a broad array of domains. However, in most of these environments, the users must write models in certain programming languages, such as Java and C++, with predefined libraries. This therefore results in their relatively limited adoption within the industry [5].

Recently, several DEVS-based graphical modeling approaches [5-7] have been proposed to make DEVS more accessible to a wider community. A DEVS graph [6], which permits the users to write DEVS models using state machines, was proposed by Christen et al., and was later adopted in graphical modeling editors of CD++Builder [5]. Traoré proposed a DEVS-driven modeling language [7] that considers functional, dynamical, and static aspects of systems in a graphical way. However, they have addressed only the graphical modeling of discrete systems, while most large-scale systems, such as cyber-physical systems (CPS) [8,9], involve hybrid processes combining both continuous and discrete phenomena.

---

* This work was supported by the IT R&D Program of MKE/KEIT [10035708, "The Development of CPS (Cyber-Physical Systems) Core Technologies for High Confidential Autonomic Control Software"].

© Springer-Verlag Berlin Heidelberg 2011
This paper proposes a visual modeling language (DVML) that enables graphical modeling of hybrid systems, based on discrete event and differential equation systems (DEV&DESS) formalism developed by Praehofer [10]. In DVML, a hybrid system can be defined using two different kinds of diagrams: a basic model diagram and coupled model diagram. The hybrid behavior of systems can be graphically defined using basic model diagrams, while coupled model diagrams can describe the structure of the systems. For hybrid systems modeling, by using DVML, the users are no longer required to have programming experience. A prototype of a DVML-based modeling and simulation environment is introduced at the end of this paper.

2 DEVS-Based Visual Modeling Language (DVML)

This section presents the proposed modeling language, DVML, in detail.

2.1 Basic Model Diagram (BMD)

DVML specifies a basic DEV&DESS model using a basic model diagram (BMD). A BMD is depicted as an empty rectangle with the name of the model at the top left. All elements that belong to a BMD are shown within the rectangle.

Each basic model interacts with its external environment by passing continuous values or discrete events through its I/O ports. Each port is represented by a small rectangle drawn on the border of the BMD, with the textual notation for the ports. The notation is defined by the following Backus-Naur Form (BNF) expression:

\[
<port>::=''[<port-type>]''<port-name>':'<data-type>
\]

\[
<port-type>::=''[C]'|'[E]
\]

‘[C]’ is specified for continuous value ports. For a discrete event port, ‘[E]’ is specified and its value can be either a predefined event (an element of <data-type>) or ∅ (nonevent). An input port and an output port are drawn as a small empty rectangle and a small filled rectangle, respectively. A discrete event port can be accessed only within transitions, while continuous value ports can be accessed at any time.

In DVML, the specification of the behavior of a basic model is organized around the phase variable in an enumerated type of programming language. The phase variable denotes some type of representative state that the model remains in. In each phase, an enumerator in the variable is rendered as a rounded rectangle along with its name. The initial phase is depicted as a small filled circle.

Other state variables are denoted by text strings in the notation defined by the following BNF expression:

\[
<state-variable>::=''[<variable-type>]''<variable-name>':'<data-type>
\]

\[
<variable-type>::=''[C]'|'[D]
\]

‘[C]’ is specified for continuous state variables, while ‘[D]’ is specified for discrete state variables, which are updated by discrete events.