Towards Dynamic Structure Hybrid DEVS for Scientific and Technical Computing Environments

C. Deatcu, T. Pawletta, Hochschule Wismar – Univ. of Technology, Business & Design, Germany

SNE Simulation Notes Europe SNE 19(3-4), 2009, 75-78, doi: 10.11128/sne.19.sn.09959

The established Discrete Event System Specification (DEVS) offers opportunities to comprehensively describe discrete event systems. In this paper the classic DEVS approach is extended with specifications and methods for continuous and variable structure modelling to hybrid models with a variable modular, hierarchical structure. To let engineers benefit from these powerful modelling instruments, they are integrated into the well-accepted and popular scientific and technical computing environment (SCE) Matlab. Furthermore, integration with other computing methods provided by the chosen SCE is obvious and can be accomplished. We appreciate DEVS based algorithms for modelling and simulation in SCEs as a complement and addition to existing tools as Simulink, Stateflow and SimEvents for engineering tasks such as e. g. discrete eventbased control design. Moreover, a SCE provides a prototyping environment for the evolution of the DEVS approach itself.

Introduction

Since Zeigler et al. introduced the Discrete Event System Specification (DEVS) in the seventies [8], many extensions of the DEVS formalism where designed. This paper presents an approach for dynamic structure hybrid DEVS, the DSDEVS-hybrid formalism, depicts its formal modelling and simulation concepts and gives an overview on the implementation of the DSDEVS-hybrid toolbox for Matlab.

Our modelling approach for modular hierarchical hybrid systems with structural variability at the coupled system level is based on the work in [6, 1, 4] and classic DEVS theory [9]. A dynamic structure system in DEVS context is a modular hierarchical system whose structure changes during simulation time. Hierarchical models are composed of two system types, atomic and coupled models. Coupled models consist of other coupled models and/or atomic models. The dynamic behaviour of a system is reflected in atomic models, while dynamic structure is defined at the coupled system level. Dynamic structure changes are e.g. creation, deletion and exchange of models. Hybrid means that besides discrete model fractions, continuous model parts are contained, as well.

There are two ways to approach those kinds of systems, resulting from the two general worldviews. One approach starts from the continuous modelling and simulation worldview and therefore extends a continuous model to a hybrid one. Discrete events are expressed as root-finding problems. The model is then simulated by a continuous simulation engine, i.e. it is processed by an ODE solver with discontinuity detection and localisation.

Usually, in modular hierarchical modelling and simulation environments, the model structure is flattened before execution. Hence, hierarchical structure information is partly not available during simulation time and dynamic structure behaviour needs to be elaborately modelled at atomic system level. It seems to be more promising to gain access to the problem through the second worldview, the discrete event worldview. In that case a discrete simulator rules the simulation engine and calls an ODE solver to compute continuous model fractions. Among descriptions for discrete event system models and their simulators, we have chosen and enhanced the Discrete Event System Specification (DEVS). In contrast to other ongoing research, e.g. integration of DEVS into the Modelica language [7] that employs DEVS for the description of only the discrete part of hybrid models, the described approach takes DEVS as the basis. DEVS itself and particularly its related simulator concepts are extended with hybrid and at the same time dynamic structure features.

1 Formal modelling concept

An overview of the formalisms that underpin and extend DEVS theory is given by Zeigler et al. in [9]. One of several extensions of the basic DEVS formalism is the hybrid DEVS formalism presented by Praehofer [6]. Another approach for hybrid DEVS modelling was introduced by Kofman [3, 2], who proposes quantisation of the state variables instead of time discretisation to approximate differential equations. The suitability for dynamic structure modelling of this approach is not followed up in this paper.

1.1 Atomic model specification

Praehofer [6] defined a hybrid atomic system by the tuple

$$A_{\text{hybrid}} = (X, Y, S, f, c_{\text{se}}, \lambda_c, \delta_{\text{x\&s}}, \delta_{\text{int}}, \lambda_{\text{d}}, ta) \quad (1)$$

where X, Y, and S specify the set of inputs, outputs and states which may be continuous or discrete. Continuous dynamics are mapped by the rate of change function f and the continuous output function λ_{c}. Discrete events are internal, external and state events. State event conditions are defined using the state event condition function c_{se}. External events and state events induce state transitions using the function $\delta_{\text{x\&s}}$. Internal events activate the discrete output function λ_{d} and also the internal state transition function δ int . After each discrete state transition internal events are rescheduled by the time advance function ta. Local structural changes of the continuous dynamics can be modelled by structuring the dynamic description using logic variables inside the rate of change function f, the state event condition function c_{se} and the continuous output function λ_{c} . This definition for hybrid atomic DEVS fits for being combined with a coupled system definition for dynamic structure models. Hybrid behaviour is defined on atomic system level, while dynamic structure is modelled at coupled system level.

1.2 Coupled model specification

In order to allow dynamic structure behaviour, e.g. the creation, deletion and exchange of submodels, additional data structures to represent this dynamic need to be established. Barros [1] proposes to add a special atomic model called network executive to each coupled model which holds the structure information and describes the structure dynamics. In contrast to this approach we favour the extension of the classic DEVS coupled model definition in way that allows to hold structure information directly in the coupled model. The specification of a classic DEVS coupled model [6, 9] is as follows:

$$N = (X_N, Y_N, D, \{M_d | d \in D\}, EIC, EOC, IC, Select) \quad (2)$$

where X_N is the set of input values and Y_N is the set of output values. The index N stands for network model as synonym for coupled model. D is the set of the component names, while M_d represents a dynamic subsystem. The property of closure under coupling, which is ensured for classic coupled DEVS, enables the representation of every coupled DEVS as an atomic DEVS. Thus, dynamic subsystems may be other coupled DEVS models or atomic DEVS models. EIC, EOC and IC define the coupling relations. The external coupling relation EIC connects external inputs to component inputs, the external output coupling EOC connects component outputs to external outputs and the internal coupling IC defines connections among components, i.e. component outputs are connected to component inputs. For couplings no direct feedback loops are allowed. Finally, $Select$ acts as a special function to prioritise one subsystem in case of simultaneous internal events in subsystems.

To allow structure variability, some extensions of this coupled system's definition have to be introduced. In the context of this work, possible structural changes at the coupled system level are

- Creation, Cloning, Deletion and
- Replacement of atomic or coupled subsystems,
- Their movement between coupled systems and
- Changes of couplings between system components.

We call the actual composition of a subsystem set and its coupling relations the structure state. A dynamic structure DEVS can have different structure states $s_0, s_1, \ldots, s_n \in S_N$. Furthermore, structure dynamics information, e.g. the number and kind of structure changes already achieved, needs to be stored. The set of structural variables H_N holds this information. For a dynamic structure coupled DEVS the $Select$ function can depend on the structure state and structure dynamics information. Consequently, we define the set of sequential structure states S_N of a dynamic structure coupled DEVS as:

$$S_N = H_N \times D \times \{M_d | d \in D\} \times \times EOC \times EIC \times IC \times Select \quad (3)$$

This set of sequential structure states extends the formal definition of classic coupled DEVS without structure variability to the dynamic structure DEVS definition. We define a dynamic structure DEVS as follows:

$$N_{\text{dyn}} = (X_N, Y_N, \{d_N\}, S_N, \delta N_{\text{x\&s}}, \delta N_{\text{int}}, \lambda N_d, taN) \quad (4)$$

Note that coupling information as well as $Select$ rules are now capsuled in the set of structure states S_N. The name of the coupled system is stored in d_N. Furthermore, a dynamic structure hybrid DEVS implies the functions $\delta N_{\text{x\&s}}$, δN_{int}, λN_d and taN. The transition, output and time advance functions until now were defined for atomic hybrid DEVS only. These functions provide operations similar but not identical to those for atomic systems.

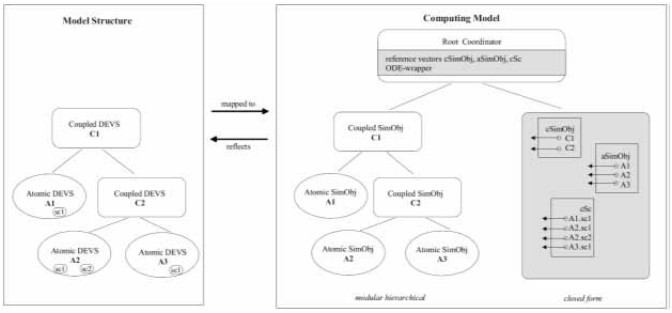

Figure 1. Mapping elements of a model structure to simulation objects of a computing model

In analogy to event-oriented dynamic behaviour of classic atomic DEVS systems, dynamic structure changes in coupled DEVS are induced by events. Relevant events are external, internal or state events. If an external event occurs, it will be sent to the affected subsystems, as known from classic DEVS. If it influences the structure dynamics of the coupled system, its state transition function $\delta N_{\text{x\&s}}$ is executed. After that the time advance function taN is called to recalculate the time until the next internal event. State events that affect structure can be caused by output events of subsystems or threshold events of (i) continuous outputs of subsystems, (ii) continuous inputs of the coupled system or (iii) structure related states of the set H_N.

For state events first $\delta N_{\text{x\&s}}$ is executed, and then taN is called. The time advance function taN for coupled DEVS also schedules events triggered by the internal structure of a coupled DEVS. The structure changes to be accomplished are specified with the structure state transition function δN_{int}. For the generation of structure related output events caused by internal events, the discrete output function λN_d is introduced. Presented specifications for atomic and coupled DEVS models together with a new simulation concept form the DSDEVS-hybrid formalism. The formal approach and its application on a real engineering system are described in detail in [5].

2 Simulation concept

In dynamic structure as well as static structure hybrid DEVS formalisms and associated simulator concepts the continuous part of the model causes events to occur in the DEVS part. The model is simulated by a modified discrete event simulation engine which calls an ODE solver during simulation cycles. Structure information of the hybrid modular hierarchical model remains available during simulation time. Thus, the design and simulation of dynamic structure hybrid models becomes imaginable. Computation algorithms for modular hierarchical DEVS models including dynamic structure and hybrid system extensions were established in [9, 6, 1, 5]. The key idea is to map a model specification to interacting program objects to reflect the system components and their coupling relations. This means for each part of the hierarchical model a program object exists which exclusively handles the dynamics, i.e. the simulation of this model part. These program objects are referred to as simulation objects of the computing model. On top of the hierarchically organised computing model the root coordinator initiates and controls the simulation cycles. Figure 1 illustrates the relations between the specified model and resulting program objects. The regions highlighted in grey are not part of classic DEVS formalisms, but extensions introduced with the DSDEVS-hybrid formalism.

Until now, problems arised for the effective calculation of continuous model parts if they are distributed over different program objects. Current approaches work with the Euler method and do not support the use of other ODE solvers. However, engineers ask for advanced ODE solvers with implicit integration methods, predictor/ corrector integration methods and automatic step width control to solve e.g. stiff systems. The proposed DSDEVS-hybrid formalism based on [5] comprises new data structures and methods which automatically generate the description of the continuous model equations and continuous state vectors of all model components in a closed form. This closed description is prerequisite for the efficient use of advanced ODE solver methods. To achieve a closed description we make use of wrapper concepts. On top of the simulation engine the root coordinator is extended by the ODE-wrapper method. The ODE-wrapper method allows the closed model representation by using additional data structures. These data structures hold by the root coordinator are the vector of references to all continuous state variables *cSc* and the vectors *cSimObj* and *aSimObj* filled with references to all atomic and coupled models. These references provide a dynamic representation of the modular hierarchical model in the required closed form. In Figure 1 data structures and newly introduced methods are highlighted in grey. Taking benefit of the wrapper concept leads to possibilities for defining interfaces to advanced ODE solvers which are e.g. provided by programmable scientific and technical computing environments (SCEs).

3 Implementation of the DSDEVS-Hybrid Toolbox for Matlab

Unlike other modelling methodologies for discrete event systems such as Petri nets or state charts, DEVS formalisms have not been widely accepted by engineers, although they are to be seen as powerful tools to solve engineering problems. To help eliminating this lack of acceptance we propose the employment of SCEs. Since engineers, unlike scientists, are rather familiar with the use of SCEs such as Matlab than of high level programming language simulation libraries, the integration of DEVS algorithms into those environments is overdue. Furthermore, SCEs provide a large number of predefined advanced ODE solvers which can be involved to compute the continuous parts of hybrid models. Our research aims to integrate advanced DEVS algorithms into SCEs and to take benefit of combination with other computing methods and toolboxes, e.g. for distributed and parallel computing, HLA modelling and for simulation based reactive control of discrete event systems. Moreover, a SCE provides a prototyping environment for the evolution of the DEVS approach itself.

For Matlab a prototype implementation of dynamic structure hybrid DEVS modelling and simulation exists which is named *DSDEVS-hybrid toolbox*. Practicability of the formal approach is therewith proofed and verified. Major class definitions for the Matlab DSDEVS-hybrid toolbox reflecting the formal definitions for dynamic structure hybrid atomic and coupled DEVS models can be found in [4].

4 Conclusions

The described modelling and simulation approach extends and brings together DEVS-based formalisms for dynamic structure and hybrid modelling. The improvements of the simulation engine by employing the ODEwrapper method avoids from the need to flatten the model before execution. Hence, structure information remains available and dynamic structure modifications are possible. Nevertheless, the closed form required to take benefit of advanced ODE solvers can be provided. Thus, the proposed formalism is applicable to complex engineering problems. To enlarge application field, further research will add ports and parallel extensions to DSDEVS-hybrid. By integrating DSDEVS-hybrid into SCEs, engineers are encouraged to break new ground in modelling and simulation while staying in their familiar software environment.

Current toolbox implementation is done with previous Matlab releases, where supplied object oriented programming features were rather poor and leaded to a complex file hierarchy. To define a new class, the programmer needed to establish a directory for the class, where all M-files which included the methods for the class were collected. For each class method a separate M-file had to be created so that class definitions became complicated and hard to follow up. Since 2008, Matlab offers enhanced features for object oriented programming. So, reimplementation with taking benefit of these features is obvious.

References

[1] Barros, F. J.: *The Dynamic Structure Discrete Event System Specification Formalism.* Transactions of the SCS International, 1996, 13(1), 35 – 46.

[2] Cellier, F. E. and Kofman, E.: *Continuous System Simulation.* Springer Pub., 2006

[3] Kofman, E.: *Discrete Event Simulation of Hybrid Systems.* SIAM Journal on Scientific Computing, 2004, 25(5), 1771-1797

[4] Pawletta, T., Deatcu, C., Pawletta, S., Hagendorf, O. and Colquhoun, G.: *DEVS-Based Modeling and Simulation in Scientific and Technical Computing Environments*. In: Proceedings of SpringSim 2006 (DEVS Symposium), Huntsville, AL, USA, 2006, 151 - 158

[5] Pawletta, T., Lampe, B., Pawletta, S. and Drewelow, W.: *A DEVS Based Approach for Modeling and Simulation of Hybrid Variable Structure Systems.* Lecture Notes in Control and Information Science (Eds.:S. Engel et.al.), Springer Pub., 2002, Vol. 279, 107 – 130

[6] Praehofer, H.: *System Theoretic Foundations for Combined Discrete-Continuous System Simulation.* PhD thesis, VWGÖ, Vienna, 1992

[7] Sanz, V., Urquia, A. and Dormido, S.: *Introducing Messages in Modelica for Facilitating Discrete-Event System Modeling.* In: Proc. 2nd Int. Workshop on Equation-Based Object-Oriented Languages and Tools, Paphos, Cyprus, 2008, 83 – 93

[8] Zeigler, B. P., Kim, T. G. and Praehofer, H.: *Theory of Modeling and Simulation, 1st Edition.* Academic Press, 1976.

[9] Zeigler, B. P., Kim, T. G. and Praehofer, H.: *Theory of Modeling and Simulation, 2nd Edition.* Academic Press, 2000.

Corresponding author: Christina Deatcu,
Univ. Applied Sciences Wismar
RG Computational Engineering & Automation
PF 1210, D-23952 Wismar, Germany
christina.deatcu@hs-wismar.de

Received & Accepted: MATHMOD 2009
Revised: September 15, 2009
Accepted: November 3, 2009