Decentralized Approach for Efficient Simulation of Devs Models

Romain Franceschini, Paul-Antoine Bisgambiglia

DECENTRALIZED APPROACH FOR EFFICIENT SIMULATION OF DEVS MODELS

Romain Franceschini, Paul-Antoine Bisgambiglia, and Paul Antoine Bisgambiglia

University of Corsica, UMR SPE 6134 CNRS, UMS Stella Mare 3460, TIC team, Campus Grimaldi, 20250 Corti
{r.franceschini, bisgambiglia, bisgambi}@univ-corse.fr

Abstract. This paper proposes to improve simulation efficiency of DEVS models based on the classical Discrete Event system Specification (DEVS) formalism by reducing the number of messages exchanged between simulators. We propose three changes: hierarchical modeling tree *flattening* based on closure under coupling, *direct coupling* and *decentralized scheduling*. The main idea is to relieve coordinators by giving to simulators more tasks to process.

1 INTRODUCTION

The study of production systems necessitates the development of specific tools. Discrete EVent system Specification formalism [Zeigler et al., 2000] is an expressive, open and flexible formalism that can be extended. Recent studies [Vangheluwe, 2000, Zeigler, 2003], have shown that DEVS formalism may be called multi-formalism because, due to its open nature, it allows the encapsulation of other modeling formalisms to meet specific applications requirements. From a performance perspective, the formalism can be improved as it does not scale well with a large number of models to simulate. At hardware level, it is possible to scale vertically by increasing power of machines, and horizontally by parallelizing [Chow and Zeigler, 1994] simulations, with a cost [Balakrishnan et al., 1997, Chow et al., 1994, Glinsky and Wainer, 2006]. At software level, we can work on algorithms efficiency by reducing their complexity.

In the DEVS formalism, the model hierarchy suggests that each evolution of a model state can produce a message, which traverse all the hierarchy up to the root of the tree. The number of messages is therefore proportional to the output of the models, the number of models, and the level of the hierarchy. In certain cases, this can affect and raise simulation execution time. Previous works already proposed different approaches to improves simulation efficiency. We can cite: parallelization approach [Chow et al., 1994, Balakrishnan et al., 1997, Kim et al., 2000, Glinsky and Wainer, 2006, Zacharewicz and Hamri, 2007, Jafer and Wainer, 2009]; distribution [Kim et al., 2000, Liu, 2006, Zacharewicz and Hamri, 2007]; and software approaches. These last approaches improve simulation time by getting rid of the hierarchical structure [Jafer and Wainer, 2010, Jafer and Wainer, 2009, Lowry et al., 2000, Zacharewicz and Hamri, 2007], and suggests to use direct coupling between models [Chen and Vangheluwe, 2010, Muzy and Nutaro, 2005]. The purpose of this article is to propose modifications to the simulation algorithms to reduce significantly the number of exchanged messages between components.

Currently, our approach is based on the classical DEVS formalism. The rest of the paper is organized as follows: first part, we present the DEVS formalism. In the second part, we detail the modifications we introduce to the classical DEVS formalism. In the last part, we present simulation results using our "decentralized" simulation approach.

2 BACKGROUND

DEVS [Zeigler, 2003] allows representing any system whose input/output behavior can be described with a sequence of events. It allows defining hierarchical modular models with two distinct types: atomic (behavioral) and coupled (structural) models. The first describes the autonomous behavior of a discrete-event system; the last one is composed of sub-models, each of them being an atomic or a coupled model. Formally, an atomic model is described by: $\langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$, and a DEVS coupled model is described by $\langle X, Y, D, \{M_d \mid d \in D\}, EIC, EOC, IC, Select \rangle$.

We use the DEVS formalism because of its openness and extensibility. It offers both a formal framework to define models and a flexible implementation in object-oriented programming. It allows modeling all types of systems. In some cases, depending on the system, the simulation can be very time consuming. To explain the excess messages must detail the simulation part [Jafer and Wainer, 2009]. There are many works that aim to accelerate the simulation. We can cite [Glinsky and Wainer, 2006, Hu and Zeigler, 2004, Jafer and Wainer, 2009, Jafer et al., 2013, Lee and Kim, 2003, Liu and Wainer, 2012, Muzy and Nutaro, 2005, Wainer and Giambiasi, 2001, Zacharewicz et al., 2010]. Some of these solutions propose to flatten the hierarchy of models in order to reduce communication overhead between models.

This is achieved by simplifying the underlying simulator structure, while keeping the same model definition and preserving the separation between model and simulator. There are two advantages to using a so-called flat structure: reduce exchanges of messages and simplify the simulation tree. This simplification is often used to allow parallelize or distributed simulations. These many works have shown that flat simulators outperform hierarchical ones significantly. They have also showed that although the hierarchical simulator presented in [Glinsky and Wainer, 2006, Zacharewicz et al., 2010] reduced the number of messages by introducing two specialized DEVS coordinators, the communication overhead was still high in some cases. Others propose modifications simulation algorithms to parallelize and/or distribute computations (outsource). We propose to improve the simulator structure to accelerate simulation time. Our approach to accelerate simulations is not based on outsourcing the computations, but on three items: flat structure, direct coupling and decentralized scheduling.

3 OUR APPROACH

The objective of this work is not to provide a comparison with other approaches that are based on parallelization or distribution; we propose algorithms to improve the classical DEVS formalism. The aim of our modifications is to simplify the DEVS formalism, in order to make it more effective and faster. To reduce the number of exchanged messages

between DEVS components, we propose three changes while remaining in compliance with the universal properties of DEVS, such as closure under coupling.

3.1 Local schedule

In order to avoid message overhead, we propose to avoid dispatching *-messages when possible, which we will call local or decentralized schedule. The purpose of this modification is to make the simulator more autonomous and to simplify the task of flat coordinator. Right after processing an *-message, a simulator checks if it is the next scheduled simulator by its parent and if there is no other simulator scheduled at the same time. If so, then the simulator will keep control and process the new *fictive* *-message at its tn.

3.2 Direct coupling

Message generation in the DEVS formalism is caused by message routing, specifically routing induced by the hierarchical structure of the formalism. For example, a component C1 of level H2 cannot communicate directly with a component C2 of the same level (H2). This is the case for all components. Messages must always be propagated to the parent, in H1 or H0 level. This hierarchy is a source of communication too. The fact of not being able to communicate directly with a component of the same level is a problem. We propose to add a list of couplings in simulators as a state variable. The simulators know their coupling, that is to say, the components to which they are connected, and with whom they should communicate. This list of decentralized coupling has been added to simulators.

3.3 Flattening architecture and direct connection

The hierarchy flattening, also called direct connection by [Chen and Vangheluwe, 2010], is not new and has become a key to improve simulation time. The property of closure under coupling demonstrated in [Zeigler et al., 2000] implies that any coupled DEVS model offers the same behavior as a resultant atomic model, which allows to delete all coupled models in the hierarchy. In the hierarchical structure proposed in the DEVS formalism, a root coordinator is placed on top with a coordinator just below (H0 level). To flatten the simulation architecture, all the coordinators below the H0 level are deleted. Other works already offer this mechanism [Jafer and Wainer, 2009, Zacharewicz and Hamri, 2007], usually in order to parallelize and distribute the simulation. Our goal is to make the simulator standalone by removing redundant communications. We still keep the top-most coordinator, positioned just below the root. It gives an execution order to simulators. It has a schedule with an event number equal to the number of simulators. Now that we flattened the hierarchy, the top-most coordinator still coordinates its components. A component that generates an y-message still pass by its parent, which could be avoided with direct coupling.

3.4 Algorithms

We are now going to present algorithms for the modifications we propose, based on the classical DEVS simulation algorithms defined in [Zeigler et al., 2000].

Decentralized simulator tend to reduce the number of messages generated during a simulation in two different ways: (1) by allowing a simulator to communicate directly another component of its parent whenever possible and (2) by keeping control of *-messages whenever possible.

Listing 1.1. Decentralized simulator algorithm

```
1  variables:
2    indirect-couplings = all parent ICs where oport host == self and parent
         EOC doesn't include oport
3
4  when receive (*, t) message do
5    do
6      raise synchronisation error if t != tn
7
8      y = model.lambda()
9
10     for each (y, oport)
11       couplings = indirect-couplings(oport)
12       if couplings are empty and parent EOC doesn't include oport
13         send (y, t) to parent
14       else if couplings are not empty
15         for each (iport, v) in couplings
16           send (x, t, v) to iport host
17         end
18       end
19     end
20
21     model.internal_transition()
22
23     tl = t
24     tn = t + model.ta()
25
26     t = tn
27   end while tn == parent.min_tn and parent.imminent.count < 2 and tn !=
       end-of-simulation
28 end
```

We achieve direct coupling by introducing indirect couplings to the simulator variables. Indirect couplings represent the direct route to another component of the parent. They are all IC of the parent involving one of the output port of the simulator model, excluding output ports involved in a EOC of the parent coordinator. As Listing 1.1 shows, when an output is generated by the model, an y-message has to be dispatched to the parent coordinator only if no indirect coupling exists and that the port is not involved in a EOC. If that is not the case, an x-message is directly sent to each indirect coupling recipient.

To avoid to return control from *-message, a simulator checks at the end of *-message processing if it is the next scheduled message by its parent. Then, if it is the only scheduled model at that time, it is not necessary to the parent to call the $Select$ method. In that case, the message time is set to the simulator tn and the *-message processing starts again unless we reached the end of the simulation.

4 RESULTS

The suggested approach allows to reduce the complexity of the simulation algorithms. We still have to demonstrate through some examples that this is expressed by a major reduction of the number of messages exchanged. We propose to present the results obtained with a [Wainer et al., 2011] benchmark. DEVStone allows to evaluate the performance of DEVS-based simulators. It generates a suite of model with varied structure and behavior automatically. The test environment is based on a Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz, 8 GB (2 x DDR3 - 1600 MHz) of RAM, APPLE SSD SM128E hard drive, running on OSX 10.9.3. Software used for the benchmarking is DEVS-Ruby [Franceschini et al., 2014] (without C extensions enabled) running on the Ruby 2.1.2 VM. DEVS-Ruby is a DEVS-based simulation framework implemented with the Ruby language.

4.1 Simulation results

Table 1 shows the total number of exchanged messages along with the CPU time of a simulation for each of the three approaches. The DEVstone model is parameterized with a *depth* between 3 and 9, a *width* from 5 to 15, with *HO* models type, a δ_{int} transition time of 1ms, and a δ_{ext} transition time of 0.1ms.

Table 1. Number of exchanged messages and CPU time for each approach using DEVStone

Depth	**Width**	**Approach**					
		Classic		**Flat**		**Decentralized**	
		Messages	*CPU time(s)*	*Messages*	*CPU time(s)*	*Messages*	*CPU time(s)*
3	5	132	0.027517	90	0.027047	**61**	0.027517
	10	359	0.07453	260	0.072432	**161**	0.071881
	15	722	0.172698	530	0.170959	**311**	0.148885
6	5	333	0.060773	195	0.058662	**133**	0.0571
	10	1043	0.177945	620	0.173918	**383**	0.198214
	15	2183	0.4381	1295	0.375843	**758**	0.370906
9	5	588	0.09689	300	0.093082	**205**	0.091734
	10	1952	0.285154	980	0.296403	**605**	0.276397
	15	4148	0.648915	2060	0.709434	**1205**	0.609233

Results show a major drop of scheduled messages between flat simulations and classic simulations. This is predictable because of all messages no longer sent to sub coupled models since they have been deleted from the hierarchy. Those results are very

interesting but were already obtained by previous works on hierarchy flattening. However, decentralized simulation offers very encouraging results since we can observe an additional message drop. We obtain this by reducing the number of scheduled *-messages and by avoiding to each atomic model that produces an y-message to pass by its parent coordinator by dispatching directly an x-message to the recipient.

Although we significantly reduce the number of messages and that CPU times shows slightly better results with our approach, the difference is not as impressive as the number of messages. In our case, this is due to the naiveness of the sorted list-based scheduler which is used for now in DEVS-Ruby. Indeed, the hierarchy flattening increase the number of atomic models to handle by the scheduler of the last present coordinator. Moreover, the *HO* type of coupling in DEVStone involves many collisions, which is a stress condition for the scheduler.

5 CONCLUSIONS

In this article, we presented an approach that aims to reduce the number of exchanged messages in the classic DEVS formalism. To reduce the number of messages exchanged, we propose to expand the role of simulators. Indeed, we propose three major changes compared to classical DEVS formalism: direct coupling, flat structure and local schedule. The goal is the decentralization of a number of tasks in order to make the simulators more autonomous, and relieve coordinators. Through these modifications the universal property of DEVS are preserved, and it is possible to couple a classical model with a decentralized model.

The results obtained with our framework are good; the number of exchanged messages is reduced by a factor of two. For complex systems with many components such as production systems, this method seems very interesting. As a future work, we plan to work on the PDEVS formalism.

ACKNOWLEDGEMENTS

The present work was supported in part by the French Ministry of Research, the Corsican Region and the CNRS.

References

Balakrishnan et al., 1997. Balakrishnan, V., Frey, P., Abu-Ghazaleh, N. B., and Wilsey, P. A. (1997). A framework for performance analysis of parallel discrete event simulators. In *Proceedings of the 29th Conference on Winter Simulation*, WSC '97, pages 429–436, Washington, DC, USA. IEEE Computer Society.

Chen and Vangheluwe, 2010. Chen, B. and Vangheluwe, H. (2010). Symbolic flattening of DEVS models. In *Proceedings of the 2010 Summer Computer Simulation Conference*, SCSC '10, pages 209–218, San Diego, CA, USA. Society for Computer Simulation International.

Chow et al., 1994. Chow, A., Zeigler, B., and Kim, D. H. (1994). Abstract simulator for the parallel DEVS formalism. In *, Proceedings of the Fifth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems, 1994. Distributed Interactive Simulation Environments*, pages 157–163.

Chow and Zeigler, 1994. Chow, A. C. H. and Zeigler, B. P. (1994). Parallel DEVS: a parallel, hierarchical, modular, modeling formalism. In *Proceedings of the 26th Conference on Winter Simulation*, WSC '94, pages 716–722, San Diego, CA, USA. Society for Computer Simulation International.

Franceschini et al., 2014. Franceschini, R., Bisgambiglia, P.-A., Bisgambiglia, P. A., and Hill, D. R. (2014). DEVS-Ruby: a Domain Specific Language for DEVS Modeling and Simulation (WIP). In *DEVS 14: Proceedings of the Symposium on Theory of Modeling & Simulation - DEVS Integrative M&S Symposium*, pages 393–398. Society for Computer Simulation International.

Glinsky and Wainer, 2006. Glinsky, E. and Wainer, G. (2006). New parallel simulation techniques of DEVS and cell-DEVS in CD++. In *Proceedings of the 39th Annual Symposium on Simulation*, ANSS '06, pages 244–251, Washington, DC, USA. IEEE Computer Society.

Hu and Zeigler, 2004. Hu, X. and Zeigler, B. P. (2004). A high performance simulation engine for large-scale cellular DEVS models. In *High Performance Computing Symposium (HPC'04)*, pages 3–8.

Jafer et al., 2013. Jafer, S., Liu, Q., and Wainer, G. (2013). Synchronization methods in parallel and distributed discrete-event simulation. *Simulation Modelling Practice and Theory*, 30:54–73.

Jafer and Wainer, 2009. Jafer, S. and Wainer, G. (2009). Flattened conservative parallel simulator for DEVS and CELL-DEVS. In *International Conference on Computational Science and Engineering, 2009. CSE '09*, volume 1, pages 443–448.

Jafer and Wainer, 2010. Jafer, S. and Wainer, G. (2010). Global lookahead management (GLM) protocol for conservative DEVS simulation. In *2010 IEEE/ACM 14th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 141–148.

Kim et al., 2000. Kim, K., Kang, W., Sagong, B., and Seo, H. (2000). Efficient distributed simulation of hierarchical DEVS models: transforming model structure into a non-hierarchical one. In *Simulation Symposium, 2000. (SS 2000) Proceedings. 33rd Annual*, pages 227–233.

Lee and Kim, 2003. Lee, W. B. and Kim, T. G. (2003). Simulation speedup for DEVS models by composition-based compilation. In SCS, editor, *Summer Computer Simulation Conference - SCS*, pages 395–400. SCS.

Liu, 2006. Liu, Q. (2006). *Distributed Optimistic Simulation Of Devs And Cell-Devs Models With Pcd++*. PhD thesis.

Liu and Wainer, 2012. Liu, Q. and Wainer, G. (2012). Multicore acceleration of discrete event system specification systems. *SIMULATION*, 88(7):801–831.

Lowry et al., 2000. Lowry, M. C., Ashenden, P. J., and Hawick, K. A. (2000). Distributed high-performance simulation using time warp and java. Technical Report DHPC-084.

Muzy and Nutaro, 2005. Muzy, A. and Nutaro, J. J. (2005). Algorithms for efficient implementations of the DEVS & DSDEVS abstract simulators. pages 273–279.

Vangheluwe, 2000. Vangheluwe, H. (2000). DEVS as a common denominator for multi-formalism hybrid systems modelling. In *IEEE International Symposium on Computer-Aided Control System Design, 2000. CACSD 2000*, pages 129–134.

Wainer et al., 2011. Wainer, G., Glinsky, E., and Gutierrez-Alcaraz, M. (2011). Studying performance of DEVS modeling and simulation environments using the DEVStone benchmark. *SIMULATION*, 87(7):555–580.

Wainer and Giambiasi, 2001. Wainer, G. A. and Giambiasi, N. (2001). Application of the cell-DEVS paradigm for cell spaces modelling and simulation. *SIMULATION*, 76(1):22–39.

Zacharewicz and Hamri, 2007. Zacharewicz, G. and Hamri, M. E.-A. (2007). Flattening g-DEVS / HLA structure for distributed simulation of workflows. In *Proceedings of AIS-CMS International modeling and simulation multiconference*, pages 11–16, Buenos Aires, Argentine.

Zacharewicz et al., 2010. Zacharewicz, G., Hamri, M. E.-A., Frydman, C., and Giambiasi, N. (2010). A generalized discrete event system (g-DEVS) flattened simulation structure: Application to high-level architecture (HLA) compliant simulation of workflow. *SIMULATION*, 86(3):181–197.

Zeigler, 2003. Zeigler, B. (2003). DEVS today: recent advances in discrete event-based information technology. In *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003*, pages 148–161.

Zeigler et al., 2000. Zeigler, B. P., Kim, T. G., and Praehofer, H. (2000). *Theory of Modeling and Simulation*. Academic Press, Inc., Orlando, FL, USA, 2nd edition.