PDEVS PROTOCOL PERFORMANCE PREDICTION USING ACTIVITY PATTERNS WITH FINITE PROBABILISTIC DEVS

Laurent Capocchi  
Jean-Francois Santucci  
SPE UMR CNRS 6134  
University of Corsica  
Campus Grimaldi  
20250 Corte, France  
{capocchi, santucci}@univ-corse.fr

Bernard P. Zeigler  
RTSync Corp  
12500 Park Potomac  
Potomac, MD  
zeigler@rtsync.com

ABSTRACT

This paper deals with PDEVS (Parallel Discrete Event System specification) protocol performance prediction using activity patterns with Finite Probabilistic DEVS. In 2015, Zeigler et. al showed that the PDEVS protocol is a simple-to-implement distributed simulation protocol that can provide speedups to match the best performance of classical algorithms for most situations. We propose to model the PDEVS protocol using a Markov Continuous Time Model (CTM) which is set up using two parameters: (i) the probability of a state giving an output according to input patterns; (ii) the rate of the coordinator’s release of imminents. These parameters are computed using Activity metrics and inserted into the PDEVS modeling scheme in order to predict the performance of a PDEVS protocol in the framework of distributed simulations. The approach is supported by the DEVSimPy and Ms4Me environments and is highlighted by a pedagogical example.

Keywords: Finite Probabilistic DEVS, Markov chain, Ms4Me, parallel simulation, PDEVS Protocol.

1 INTRODUCTION

Parallel discrete event simulation or distributed simulation has been intensively studied in recent years (Adegoke, Togo, and Traoré 2013; Bolduc and Vangheluwe 2001; Reuillon, Traoré et al. 2012; Sun and Chen 2010; Van Tendeloo and Vangheluwe 2015). Algorithmic formulation and practice of distributed simulation of large systems with discrete event formalism specification have now reached a maturity that allows us to consider them as competitive to the world’s differential equations continuous formalisms. The DEVS formalism (Discrete EEvent System Specification) (Zeigler 1976; Zeigler, Muzy, and Kofman 2018) is the most general discrete event formalism as it provides an explicit representation of time and allows one to specify models that do not necessarily have a finite number of states. In addition, this formalism supports: (i) A modeling approach (transition from real world to the world of the model) and therefore provides a clear interpretation of the simulation results in the real world (ii) An unambiguous operational semantics that allows, among others, to introduce a formal specification of mechanical simulation (conceptual simulator of atomic models, conceptual simulator of hierarchical models, conceptual models of distributed simulator). Parallel simulation has recently emerged as an indispensable tool for the study of the performance of major communication systems. Furthermore, parallelized (and distributed) discrete event simulation leads to
solve particularly difficult problems of parallel programming (synchronizing, load sharing). Two algorithms have marked the disciplinary field of distributed simulation: (i) Chandy and Misra offered a solution to this problem in the late 70s (Chandy and Misra 1979) with what is referred to as the conservative method; (ii) Jefferson offered a new solution called the optimistic method in the mid-80s (Jefferson 1985). Fujimoto in the nineties (Fujimoto 1993; Fujimoto 1999) has pointed out the difficulties inherent in using conservative and optimistic simulation algorithms and these problems have yet to be overcome. In (Zeigler, Nutaro, and Seo 2015) the authors showed that the PDEVS protocol is a simple-to-implement distributed simulation protocol that can provide speedups to match the best performance of classical algorithms for most situations and overcome the practical problems raised by Fujimoto. In this paper, our goal is to develop a methodology and tools to predict the performance of a PDEVS simulation before it is mapped to a parallel or distributed platform. To do so, we propose to model the PDEVS (Parallel DEVS) protocol using a Markov Continuous Time Model (CTM), which is a DEVS modeling scheme that leans on FP-DEVS (Finite Probabilistic DEVS) (Seo et al. 2015; Zeigler and Nutaro 2016) and currently runs in the Ms4Me environment (Zeigler and Sarjoughian 2013; Zeigler, Seo, and Kim 2013). We describe how to compute, for a given Markov CTM atomic model describing the PDEVS protocol, several parameters such as the probability of a state giving an output according to input patterns (an atomic model behavior property) and the rate of the coordinator’s release of imminents (a coupled model property). These parameters are inserted into the PDEVS modeling scheme in order to predict the performance of the PDEVS protocol in the framework of distributed simulations. Activity metrics can be used to profile DEVS models before and during the simulation (Muzy, Capocchi, and Santucci 2014; Muzy et al. 2011; Muzy and Zeigler 2012; Santucci and Capocchi 2014). The approach is supported by the DEVSimPy and Ms4Me environments, an example of collaborative software frameworks based on the same (DEVS) formalism.

The rest of the paper is organized as follows: the next section is dedicated to the background which involves the DEVS formalism, the Finite Probabilistic DEVS extension and the Ms4Me and DEVSimPy DEVS M&S software framework. Section 3 details the PDEVS protocol modeling scheme. In Section 4 we describe how to use activity patterns in the framework of PDEVS protocol modeling. Section 5 is dedicated to present the efficiency of the approach on a pedagogical example. The last section concerns the conclusions and future work.

2 BACKGROUND

2.1 DEVS Formalism

Zeigler introduced the DEVS formalism in the early 70’s for modeling discrete-event systems in a hierarchical and modular way (Zeigler, Muzy, and Kofman 2018). With DEVS, a model of a large system can be decomposed into smaller component models with coupling specification between them. DEVS formalism defines two kinds of models: (i) atomic models that represent the basic models providing specifications for the dynamics of a sub-system using transition functions; (ii) coupled models that describe how to couple several component models together to form a new model. DEVS provides an automatic simulation based on time synchronization and message propagation. Parallel DEVS (PDEVS) extends classic DEVS essentially by allowing bags of inputs to the external transition function and confluent transition function. Bags can collect inputs that are built at the same time, and process their effects in future bags. This formalism offers a solution to manage simultaneous events that have to be sequentially processed with classic DEVS.

An atomic model allows specifying the behavior of a basic element of a given system. Atomic model behavior is determined by:
Capocchi, Santucci, and Zeigler

- internal transition ($\delta_{\text{int}}$): it describes the autonomous behavior of the model and determines how the states evolve when there is no input
- external transition ($\delta_{\text{ext}}$): it describes how the model responds to input and how it will change its state
- confluent transition ($\delta_{\text{con}}$): it handles the simultaneous occurrence of an internal and external event
- output function ($\lambda$): it gives the output messages of the model that are caused by an internal event
- time advance function ($t_{\text{a}}$): it schedules autonomous changes of state; i.e., the next internal event. It determines the maximum lifetime in the current state

Coupled models meanwhile describe the system’s structure. They allow hierarchical level of view and describe the link between models. The communication between models is possible through message exchanges that represent events.

![Diagram of FPDEVS](image)

The abstract simulator of a PDEVS model is founded on maximal parallelism: multiple components are allowed to undertake at the same time an independent state transition.

### 2.2 Finite Probabilistic DEVS

Finite Probabilistic DEVS (FP-DEVS) (Zeigler and Nutaro 2016; Seo et al. 2015) is an extension of Finite Deterministic DEVS (FD-DEVS) (Zeigler and Sarjoughian 2013). FP-DEVS allows the transition out of a state to be one of a finite set of possible states where the choice is made probabilistically whereas FD-DEVS makes an internal transition from a state to a state. FP-DEVS extends the FD-DEVS specification of the internal transition table in (Zeigler and Sarjoughian 2013; Seo et al. 2015) as follows:

$$\text{IntTransTable} : \text{StateSet} \times ([0,1] \cup \text{Blank}) \rightarrow \text{StateSet}$$

For example, when a probability value ($x_1$) is used, $\text{IntTransTable}($PHASE$, x_1) = \text{PHASE}'$, where $0 \leq x_1 \leq 1$. To express FD-DEVS, $\text{IntTransTable}($PHASE$, \text{Blank}) = \text{IntTransTable}($PHASE$) = \text{PHASE}'$

The IntTransTable determines the phase of the next state in DEVS by: (i) Gathering all the entries for the current phases; (ii) If there are more than one, then creating a cumulative distribution function from their probability values (iii) Selecting one using a random number generator (iv) Set the new phase to the one selected.

The Figure 1 illustrates an example of FPDEVS which has one input (?) and two outputs (!). ‘B’ state can be changed to two states according to probability values at the internal transition stage.
2.3 Ms4Me Software

Ms4Me software (Seo et al. 2013) implements DEVS M&S with a Java computer language on the Eclipse environment (Steinberg et al. 2009).

![Figure 2: PDEVS protocol modeling scheme.](image)

In the Ms4Me software, the atomic model can be constructed from a state diagram designer which is a graphic user interface to help users visualize behaviors of the atomic model with symbols. Also, it can be created using a restricted DEVS Natural Language (DNL) whose keywords are highlighted in a DNL editor in the Ms4Me software. The atomic model component contains states, transitions, and ports (input and output). The coupled model is an instance of System Entity Structure (SES) (Zeigler, Seo, and Kim 2013) that describes a system with entities relationships and coupling information in restricted natural languages. Finite state Markov chain model classes (Pinsky 1972; Kemeny and Snell 1960), with both discrete and continuous time bases, have been implemented in Ms4Me using the above described FP-DEVS capabilities. In this paper, we are interested by the CTM (Continuous Time Markov) implementation (Zeigler and Nutaro 2016). The CTM interpretation employs the probabilities associated with internal transitions as specifications of their time advances. Thus the smaller a probability of transition is, the longer is the associated time to its next occurrence (on average) and the less likely it is to be selected as the transition to take.

2.4 DEVSimPy Framework

The activity patterns computation presented in the next section has been implemented using the DEVSimPy framework. The DEVSimPy (Capocchi et al. 2011) framework is an open source software (under GPL V.3 license) dedicated to the DEVS modeling and simulation of complex systems. The aim is to provide a Graphical User Interface (GUI) for the M&S of PyDEVS models (Bolduc and Vangheluwe 2001). PyDEVS is an Application Programming Interface (API) allowing the implementation of the DEVS formalism in Python language.
3 PDEVS PROTOCOL MODELING

In this section we present the PDEVS protocol modeling using the FP-DEVS introduced in (Zeigler and Nutaro 2016). We also introduce how to measure parameters involved in the PDEVS model and describe how using these measurements in order to estimate the parallel utilization of PDEVS.

3.1 PDEVS Protocol Model

For a coupled DEVS model, the Parallel DEVS simulation protocol is briefly reviewed below:

Until specified number of global transitions done:
Do global transition:
- For each imminent (own $t_N = global\ t_N$):
  - compute output and send it to receivers (1)
- For each active (imminent and input receiver),
  - compute state transition -(ext, int, conf)- (2)
- notify own $t_N$
- Advance global clock, global $t_N = min\ active\ tNs$

remarks: $t_N$ stands for the time of next event.

We can notice that (1) and (2) can each be executed in series or in parallel. Imminent components execute their internal state transitions simultaneously. The modeling scheme of the PDEVS protocol using the CTM DEVS extension is given in Figure 2.

Figure 2 describes the states and transitions involved in a DEVS atomic model allowing to represent the DEVS protocol. The set of states and the associated behavior is described below:

- **WaitForImminent**: Having sent the TransitionDone message to the coordinator, the simulator is waiting for the return imminent message (*-message), which will allow it to proceed to compute its next internal transition
- **Imminent**: Having received the imminent message, the simulator will transition to either the DoOutput or the DoStateTransition states, depending on probability parameters of the model
- **DoOutput**: The simulator outputs the model’s output event and goes to DoStateTransition
- **DoStateTransition**: The simulator performs the model’s internal transition unless it receives an external event, in which case it goes to the ConfluentTransition state
- **ConfluentTransition**: This represents having received an external event while ready to perform an internal transition. The internal transition performed may be different from that that would have been done in DoStateTransition, but we simplify by transitioning to the same InternalTransition state
- **ImmediateOrContinue**: This state is entered if the simulator receives an external event while waiting for the coordinator’s imminent message. Depending on the model’s probabilities, the simulator can transition to Continue or InternalTransition
- **Continue**: This state represents ignoring the input and returning to WaitForImminent
- **InternalTransition**: This state represents performing the internal transition and then sending the TransitionDone message to the coordinator

3.2 PDEVS Protocol Model Parameters Measurements for Parallel Utilization Estimation

In order to use the Markov model of the PDEVS protocol, we have to perform the following tasks for each atomic model involved in the coupled model under study:
Capocchi, Santucci, and Zeigler

- Estimate the probability of a state giving an output according to input patterns (called first parameter).
- Measure the rate of release of imminents by the coordinator (called second parameter).
- Estimate the probability to that the simulator performs the model's internal transition without receiving an external event (called third parameter).

These three parameters defined for each atomic model will be introduced in the corresponding CTM PDEVS model in order to specify: the probability of output before $\delta_{int}$ (P1 Figure 2), the probability that inputs cause $\delta_{int}$ (P2 Figure 2), the probability that the simulator performs the model's $\delta_{int}$ without receiving an external event (P3 Figure 2).

Of course we can then compute their associated probabilities (pointed out in Figure 2): the probability of $\delta_{int} (1 - P1)$, the probability that input does not cause any rescheduling $(1 - P2)$, the probability that the atomic model have received an external event while ready to perform an $\delta_{int} (1 - P3)$.

Once these parameters are computed using patterns analysis as presented in section 4, they will be used in order to estimate the parallel utilization linked with the PDEVS protocol. As depicted in Figure 2, the rate of transition to Continue from ImmediateOrContinue will be computed as $1 - P2$ while the rate of transition to InternalTransition from ImmediateOrContinue will be computed as $P2$. The rate of transition to DoOutput from Imminent will be computed as $P1$ while the rate of transition to DoStateTransition will be computed as $1 - P1$. In the same way the rate of transition to ConfluentTransition from DoStateTransition will be computed as $1 - P3$ while the rate of transition to InternalTransition from DoStateTransition will be computed as $P3$. We point out that the Parallel Utilization (PU), is the probability of activity (other than waiting for the coordinator to activate the simulator with an imminent input (i.e., *message below) and so:

$$PU = 1 - Wait\text{ForImminent}. \tag{1}$$

Let us call $p$ the rate of transition from ImmediateOrContinue to Continue. This is the inverse of the average time to compute the response to an external event. Consider the special case of "receive only" mode (Figure 2). For this case, it can be shown that the equilibrium probability for the WaitForImminent state is equal to $\frac{p}{1+2p}$. From Eqn 1, the parallel utilization in this case is equal to:

$$PU = \frac{1 + p}{1 + 2p} \tag{2}$$

This verifies intuition that the greater the average computation time, the closer the parallel utilization is to one (most of the atomic simulators are busy computing input responses). Thus, these two formulae allow to make the connection to parallel implementations and could be used for future research to verify such predictions. This demonstration of using the measurements to estimate parallel utilization will be used in section 5 that deals with the pedagogical example.

4 PDEVS PROTOCOL PERFORMANCE PREDICTION USING ACTIVITY PATTERNS

4.1 DEVS Activity Patterns Profiling

The sub-section concerns activity metrics which can be used to profile DEVS models before and during the simulation. We can distinguish static and dynamic metrics by differentiating between measuring what is actually happening (dynamic) rather than what should happen (static). We have defined a DEVS activity pattern profiling based on the dynamic quantitative activity. In (Muzy et al. 2011; Muzy and Zeigler 2012), the concept of activity is introduced for DEVS models as the number of transition functions executions.
This quantitative-activity (QA) metric consists in counting the number of state-to-state transitions in a model over some time interval. Activity metrics can be used to profile DEVS models.

Another notion which helps to perform Activity Tracking (AT) at the simulation level is the simulation time spent by the coordinator waiting for activity. This notion provides a natural number which represents a good measure of simulator activity needed to simulate a model. The DEVS activity pattern profiling includes the computation of the average simulation time that a model waits for the coordinator to give it a *message which we call the *Time metric. It also permits to compute the average simulation time (we call T metric) that it takes for a model to go from imminent to end of the $\delta_{int}$. The DEVS activity pattern profiling has been realized using the DEVSimPy framework (Muzy et al. 2014; Santucci and Capocchi 2014). Building upon definitions about activity concept given previously, DEVSimPy implements a new plug-in called "Activity Tracking" which is first aimed to further increase the affordability of activity concept and thus opening new perspectives for the use of AT in DEVS M&S. The DEVSimPy plug-in AT is generic and can be applied to any DEVSimPy model. It does not require any modification on the DEVS simulation algorithm and does not require any additional methods in DEVSimPy models to operate. It works in the following way:

1. Before the simulation, the DEVS models are scanned in a recursive way to collect all atomic models selected by the user in the plug-in interface.
2. The external, internal and output functions of all selected models are decorated with a new method aimed to introduce the AT computation of these functions. The decorator adds a new attribute to the DEVS object in a dynamic way (offered by the Python language) for each transition function. This new attribute is a dictionary with key the simulation time associated at the CPU of the tracked transition function.
3. From this dictionary, QA is measured by counting the number of its keys after the simulation. In the same way the simulation time that a model waits for the coordinator to give it a *message is also computed (difference between simulation times stored in dictionary)
4. Finally, when the simulation is over, the plug-in offers a table resuming the QA and *Time quantities for each tracked model.

In the next section we present the AT plug-in applied to PDEVS Protocol.

4.2 Activity Patterns for PDEVS Protocol Performance Prediction

The QA metric gives the number of output transition function activations with the generation of an output ($Q_{out}$) as well as the $\delta_{int}$ activations ($Q_{int}$) using a simulation process. The idea is to perform a set of pseudo-random generation of input patterns which are simulated in order to estimate the probability of a state giving an output according to inputs patterns. The rate is given by $Q_{out}$ that corresponds to the first parameter (the probability of the state giving an output). In order to compute the second parameter (the rate of release imminents) we can point out that this rate is the inverse of the average time that a model waits for the coordinator to give a *message. This time is given by the DEVSimPy AT plug-in which computes the *Time metric (section 4.1). The RIT metric (Release of Imminents Time) is then obtained as the inverse of the *Time metric. In order to compute the third parameter (the probability that the simulator performs the model’s $\delta_{int}$ without receiving an external event) we can point out that this probability can be computed as the inverse of the average time that it takes for a model to go from imminent to end of the $\delta_{int}$. This time is given by the DEVSimPy AT plug-in which computes the T metric (section 4.1). In order to compute the third parameter (the probability that the simulator performs the model’s internal transition without receiving an external event) we can point out that this probability can be computed as the inverse of the average time that it takes for a model to go from Imminent to end of the internal transition. This time is given by the
DEVSimPy AT plug-in which computes the T metric as stated in section 4.1. The Third parameter metric is then obtained as the inverse of the T metric.

5 A CASE STUDY: THE IEEE 802.3 CSMA/CD PROTOCOL

This sub-section introduces a pedagogical example allowing to validate the proposed approach. This case study concerns the IEEE 802.3 CSMA/CD (Carrier Sense, Multiple Access with Collision Detection) protocol (Kwiatkowska et al. 2004). The CSMA/CD protocol is designed for networks with a single channel and specifies the behavior of stations with the aim of minimizing simultaneous use of the channel (data collision). The basic structure of the protocol is as follows: when a station has data to send, it listens to the medium, after which, if the medium was free (no one transmitting), the station starts to send its data. On the other hand, if the medium was sensed busy, the station waits a random amount of time and then repeats this process.

5.1 The Station and Medium Models

In Figure 3, we have presented the DEVS automata model of the station model. A station starts by sending its data. If there is no collision, then, after $\lambda$ time units, the station finishes sending its data. On the other hand, if there is a collision, the station attempts to retransmit the packet where the scheduling of the retransmission is determined by a truncated binary exponential back-off process.

The delay before retransmitting is an integer number of time slots (each of length slotTime). The number of slots that the station waits after the nth transmission failure is chosen as a uniformly distributed random integer. Once this time has elapsed, if the medium appears free the station resends the data (event send), while if the medium is sensed busy (event busy) the station repeats this process.

The medium is initially ready to accept data from any station (event send). Once a station starts sending its data there is an interval of time (at most $\sigma$), representing the time it takes for a signal to propagate between the stations, in which the medium will accept data from the other station (resulting in a collision). After this interval, if the other station tries to send data it will get the busy signal (busy). When a collision occurs, there is a delay (again at most) before the stations realize there has been a collision, after which the medium will
Capocchi, Santucci, and Zeigler

become free (represented by the event cd). If the stations do not collide, then when a station finishes sending its data (event end) the medium becomes idle. The medium state automata is summarized in Figure 4. For AM2 atomic model we have to compute the three previously described parameters.

5.2 PDEVS Modeling

A DEVS coupled model involving three atomic models as shown in Figure 5 has been implemented in order to model the CSMA/CD protocol. Each of the three atomic models has its own such CTM with parameters we have been able to compute using activity patterns. In the validation process we are interested by the AM2 atomic model (Medium model) since the two AM1 models are generators (Station models).

![Figure 5: DEVS coupled model of the CSMA/CD protocol.](image)

5.3 DEVSimPy Modeling and Simulation

Figure 6 depicts the DEVSimPy model of the case study. "Station_1" and "Station_2" can be considered as the model AM1 and the "Medium_3" as the model AM2. Simulation has been performed during 1000 seconds (simulation time).

![Figure 6: The CSMA/CD protocol DEVSimPy model.](image)

Figure 7 shows the setup interface of AT plug-in in DEVSimPy for the DEVS model of the case study. Once the user has enabled the plug-in activation, he has to select both the atomic models, functions ($\delta_{ext}$, $\delta_{int}$, $\lambda$ or $t_a$) and type AT (static or dynamic) under investigation.
We have selected the "Medium_3" atomic model, the $\delta_{\text{ext}}$, $\delta_{\text{int}}$ and $\lambda$ functions and both the static and dynamic AT measures. Having selected the set of tracked models, the simulation can be run to proceed to the AT computation. The activity patterns profiling executed for the three atomic models is given in Table 1. It points out the results of the DEVSimPy AT plug-in for the three atomic models concerning $QA_{\text{int}}$, $QA_{\text{out}}$ and the *Time (average time that a model waits for the coordinator to give it the *message).

Table 1: Activity profiling results after simulation and probability of a state giving an output and RIT.

<table>
<thead>
<tr>
<th>States</th>
<th>$QA_{\text{int}}$</th>
<th>*Time(send-only)-(receive-only)</th>
<th>$QA_{\text{out}}$</th>
<th>$QA_{\text{out}}/QA_{\text{int}}$</th>
<th>RIT(send-only)-(receive-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station_1</td>
<td>32</td>
<td>5.94-53.73</td>
<td>6</td>
<td>0.5</td>
<td>0.16-0.018</td>
</tr>
<tr>
<td>Station_2</td>
<td>5</td>
<td>25.25-806</td>
<td>2</td>
<td>0.4</td>
<td>0.03-0.001</td>
</tr>
<tr>
<td>Medium_3</td>
<td>18</td>
<td>57.18-48.9</td>
<td>15</td>
<td>0.83</td>
<td>0.017-0.02</td>
</tr>
</tbody>
</table>

The $QA_{\text{int}}$, $QA_{\text{out}}$ and RIT ($\frac{1}{*Time}$) are used to compute the three parameters in the PDEVS protocol modeling scheme. The results are given in Table 1.

Once the "Medium_3" PDEVS model has been defined using the activity patterns process we are able to perform the simulation in the Ms4Me framework. We have performed the simulation using the send-only mode (no external events have occurred, Figure 2). The obtained results are described in Table 2. Table 2 points out, for all the five states, which are active in the send only mode, associated probabilities obtained after simulation. We are especially interested by the WaitForImminent state. We observe that the probability for WaitForImminent state is 0.933. So we are able to predict that $PU = 1 - 0.933 = 0.067$ using the Eqn 1 of section 3.2. The results predict how waiting for the coordinator can be a significant factor in parellization if the output is not time consuming.

Table 2: Prob. of the 5 States being active in send-only.

<table>
<thead>
<tr>
<th>States</th>
<th>DoOutput</th>
<th>DoStateTransition</th>
<th>Imminent</th>
<th>InternalTransition</th>
<th>WaitForImminent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probabilities</td>
<td>0.013</td>
<td>0.226</td>
<td>0.0158</td>
<td>0.016</td>
<td>0.993</td>
</tr>
</tbody>
</table>

In the receive mode only, using the fact that $p$ the rate of "transition to continue" can be computed using the RIT metric given by the AT DEVSimPy plug-in, we can obtain $p$ using the operation $1 - \text{RIT (receive-only)}$. For "Medium_3", $p=1-0.02=0.98$. So using Eqn 2, $PU = \frac{(1+p)}{(1+2p)} = 0.67$. We can notice that the PU in the
send only mode is smaller than the PU in the receive only mode (end of section 3.2). We can point out where this prediction can be used when dealing with modeling and simulation: (i) the PU prediction can be used at the end of the modeling phase before the simulations in order to select the use of the PDEVS protocol in order to perform simulations on multi-core machines if the PU prediction is high; (ii) if the PU prediction if low, the model design phase can be reexamined in order to obtain a better PU prediction.

6 CONCLUSION AND FUTURE WORK

In this paper we have shown how activity patterns can be used in order to elaborate a modeling scheme for the PDEVS protocol using a Markov CTM DEVS atomic model. The PDEVS protocol offers a straightforward solution to the problems raised by the conservative and optimistic algorithms (Zeigler, Nutaro, and Seo 2015). This paper described how activity patterns are used to calibrate the parameters required by the CTM Markov model. Future work will involve two aspects: (i) we plan to allow the user to perform statistical validity and to use the usual methods to assess the confidence limits, etc; (ii) we will attempt to verify that the predictions about parallel utilization using the theory are correct. These verifications will be performed using real cases of parallel and distributed implementations.

REFERENCES


**AUTHOR BIOGRAPHIES**

**LAURENT CAPOCCHI** has been an assistant professor in computer sciences at the University of Corsica (France), SPE CNRS 6134 Research Lab since 2007. His email address is capocchi@univ-corse.fr.

**JEAN-FRANCOIS SANTUCCI** has been a full professor in computer sciences at the University of Corsica (France), SPE CNRS 6134 Research Lab since 1995. He was assistant professor at the EERIE School, Nimes, France from 1987 to 1995. His email address is santucci@univ-corse.fr.

**BERNARD P. ZEIGLER** is Chief Scientist at RTSync Corp., Professor Emeritus of Electrical and Computer Engineering at the University of Arizona, Tucson and Co-Director of the Arizona Center for Integrative Modeling and Simulation. His email address is zeigler@rtsync.com.