ITM Web of Conferences **1**, 01001 (2013)
DOI: 10.1051/itmconf/ 20130101001

Implementation and Analysis of DEVS Activity-Tracking with DEVSimPy

J.F. Santucci[1,a] and L. Capocchi[1,b]

[1] *SPE UMR CNRS 6134 Laboratory, University of Corsica, Quartier Grimaldi, 2050, Corte, France.*

Abstract. This paper deals with the implementation of Activity-Tracking (AT) paradigm in the DEVSimPy environment. DEVSimPy has been developed at the SPE UMR CNRS 6134 (University of Corsica) in order to facilitate the modeling and the simulation of discrete-event systems described with the DEVS formalism. The AT paradigm is increasingly being used by DEVS modelers for example to improve structural aspects of models (modeling level) or to make simulation more efficient using distributed or parallel algorithm (simulation level). Although DEVSimPy can be used as a software providing some features reusable for the implementation of AT paradigm (simulation profiling or step-by-step simulation), the object oriented way used to develop DEVSimPy makes it a powerful tool to implement AT paradigm in a generic manner. DEVSimPy may be viewed as an experimental framework allowing the implementation of new AT techniques while opening up new perspectives in the field of activity modeling and simulation theory.

1 Introduction

Over the past decade, numerous efforts have been made to define activity in modeling and simulation [1, 2, 6, 7, 11]. The activity notion for a DEVS system is commonly referred as the number of transition functions executions [8]. Usually the activity notion is referred as QA (Quantitative-Activity). The activity concept on the other hand has been used to connect information processing and energy consumption as proposed in [7]. In this case it is called weighted activity (WA) and allows for example when counting the number of transitions to compute the weight of a transition proportionally to the time spent in state before the transition. In both cases (for the QA and WA notions) the activity is a notion defined at the modeling level but computed during the simulation. We propose to introduce activity notions at the simulation level by computing: (i) the McCabe Cyclomatic Complexity [9] of the transition functions (called in the following MCC) and (ii) the CPU time consumption corresponding to the execution of transition functions (called in the following CPU). The first one (MCC) is defined at the simulation level but computed before simulation while the second one is defined also at the simulation level but is computed during the simulation. The aim of this paper is twofold: (i) on the one hand it formally introduces the two new activity notions called MCC and CPU;

[a] J.F. Santucci is with the SPE Laboratory (UMR CNRS 6134), University of Corsica, Quartier Grimaldi, 20250, Corte, France, e-mail: santucci@univ-corse.fr

[b] L. Capocchi is with the SPE Laboratory (UMR CNRS 6134), University of Corsica, Quartier Grimaldi, 20250, Corte, France, e-mail: capocchi@univ-corse.fr

(ii) on the other hand it proposes an evolution of the DEVSimPy framework [4] involving Activity Tracking (AT) concepts - this evolution will allow to compute the following four activity notions: QA, WA, MCC and CPU. Furthermore, we will describe in a discussion part the relations that can be pointed out between the four activity concepts and a potential evolution of the DEVSimPy framework based on activity tracking-driven simulation. This framework is being developed at University of Corsica using the Python language. DEVSimPy (http://code.google.com/p/devsimpy/) is an open source project under GPL V3 license and the SPE research laboratory team supports its development. It uses the wxPython User Interface library, the Python Language and the PythonDEVS API [3]. The DEVSimPy aims to facilitate the modeling and the simulation of dynamic systems described with the DEVS formalism [12]. Currently DEVSimPy can be used as an AT software providing some features as simulation profiling or step-by-step simulations. Furthermore, DEVSimPy has been developed in an object oriented way using as much as possible design patterns. This feature allows an easy introduction of new concepts or functionalities in order to extend the DEVSimPy framework towards a real AT software for DEVS systems. The rest of this paper is organized as follows: the next part introduces the four activity notions. QA and WA already defined in the literature and MCC and CPU which are defined in this paper. Section 3 deals with the implementation of these four activity notions in the DEVSimPy framework. In section 4 we propose an analysis and a discussion about the comparison between QA, WA, MCC, and CPU and about a potential evolution of the DEVSimPy framework based on an activity-driven simulation approach. Finally a conclusion gives the perspectives of the work.

2 The Activity-Tracking Paradigm

It is nowadays widely accepted that the AT paradigm emerges naturally from a discrete-event system and the activity of a DEVS [12] system can be tracked using two approaches: considering the modeling level and/or the simulation level [6]. Figure 1 shows the position of the AT in these two levels.

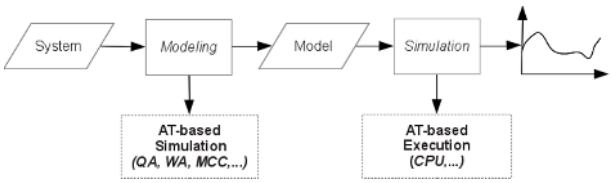

Figure 1. AT paradigm in the modeling and simulation part.

From the modeling level, tracking the activity of a model can be considered as :

- counting the number of state-to-state transitions in a model over some time interval - commonly referred to as Quantitative-Activity (QA) [8],
- counting the number of weighted state transitions over some time interval - commonly referred to as Weighted-Activity (WA) [7],
- measuring the McCabe Cyclomatic Complexity (MCC) [9] of transition functions - which is a new metrics for activity-tracking introduced in this paper.

All of these metrics can be used and combined in order to reduce message exchanges and computation at modeling level [6]. On the MCC, we explain in this paper how it can provides an interesting measure allowing for the proper identification of models which will have a large execution time.

At the simulation level, tracking the activity of the model is mainly focused on the measure of the execution time of the model or the execution time of simulation algorithms. An efficient approach to

achieve the AT at the simulation level is to consider the measure of the time spent on the processor running the code of the transition functions and the code of the simulation algorithm - typically called 'user CPU time'.

In order to understand the AT paradigm, the definition of activity must be introduced.

2.1 Quantitative Activity Definition

In [8], the authors define the Quantitative-Activity (QA) of a system as "the number of discrete-events received by the system, over a simulation time period.". According to [7], a measure of activity can be considered as a measure of information processing by counting over some time interval the number of state-to-state transitions in a model. The QA is a notion defined at the modeling level but quantified (tracked) during the simulation.

In [10], the authors give the following definition of the total activity for an atomic DEVS model N in a simulation time interval T:

$$A_N = A_{int} + A_{ext} \tag{1}$$

A_{ext} and A_{int} are resp. the external activity and the internal activity. The external (resp. internal) activity is defined as a natural number equal to the sum of DEVS external (resp. internal) transitions δ_{ext} (resp. δ_{int}) execution. Always in [7], the activity A_M of a coupled DEVS model M is defined as the sum of the total activity of its atomic model A_i included in D:

$$A_M = \sum_{i \in D} A_i \tag{2}$$

Let us consider the definition of activity given in equation 1 (resp. equation 2) as the definition of the Quantitative-Activity for an atomic (resp. coupled) model.

2.2 Weighted Activity Definition

In [7], the authors propose to link energy consumption and information processing of a system by adding a relative weight to state transition defined by the modeler through the implementation of a weighted function. According to [7] , Weighted-Activity (WA) is the sum of weighted state transitions over some time interval T:

$$A(T) = \frac{n_{ext}}{T} + \frac{n_{int}}{T} \tag{3}$$

where n_{ext} (resp. n_{int}) is the external (resp. internal) transition weighted activity defined as follow:

$$n_{ext} = n_{ext} + wt_{ext} \tag{4a}$$

$$n_{int} = n_{int} + wt_{int} \tag{4b}$$

where wt_{ext} (resp. wt_{int}) is the external (resp. internal) transition weighting function which must be implemented by the modelers. Such as for the QA notion, the WA is a notion defined at the modeling level but quantified during the simulation.

An example of the implementation of these function has been given in [7] by allocating the weight of a transition proportionally to the time spent in state before the transition. As it has been defined for the Quantitative-Activity, the Weighted-Activity for a coupled model M is equal to the sum of the Weighted-Activity of all atomic models included in D (the set of references to lower level components):

$$A_M(T) = \sum_{i \in D} A_i(T) \tag{5}$$

In order to implement an Activity-Tracking concept based on Weighted-Activity for DEVS model, the two transition weighting functions (see equations 4) must be: (i) implemented at modeling level by the modeler - (ii) introduced at the simulation level in the DEVS simulator of atomic model and invoked just before a state transition (internal or external).

2.3 McCabe Cyclomatic Complexity and CPU Notions

We introduce in this sub-section two new activity metrics: McCabe Cyclomatic Complexity (MCC) and user CPU time.

Basically, MCC is a measure used to evaluate the complexity of a program based on the number of repetitive cycles or loops included in this program [9]. It is employed for software quality control as well as to determine the number of testing procedures. However, it may be used to predict the execution time consumption of functions or methods of an object. Indeed, if the MCC of a function is high, there is a significant probability that the time spent in the execution of this function will be high. Under these circumstances, MCC can play a determining role in Tracking-Activity. It is possible to imagine that the MCC enabling to anticipate the activity tracking of model without simulate this one.

Another notion which helps to perform AT at the simulation level is the CPU time spent on the processor running the code of the model behavior (transition functions) and the code of the simulation algorithm (depending on the chosen world view). This notion commonly called 'user CPU time', provides a natural number which represents a good measure of hardware activity needed to simulate a model. With the value of measure, improving the simulation method becomes possible. An interesting proposition is the use of an activity load balancing algorithm during the simulation which relocate the models presenting the largest activity (in term of CPU) on a special process.

3 Activity-Tracking with DEVSimPy

DEVSimPy is an open source project initiated by the Modeling and Simulation team of the SPE (Sciences Pour l'Environnement) laboratory (University of Corsica). The aim of this software is to provide developers a collaborative modeling and simulation (M&S) framework in the Python language [4]. It uses the wxPython library which is a blending of the wxWidgets C++ class library with Python. The DEVSimPy M&S kernel is based on the PythonDEVS [3] API which offers a consistent and coherent set of classes in order to construct a modular system and to achieve its hierarchical simulation. The philosophy of DEVSimPy is to be an open, extensible and participative environment for the developers. A plugin manager is proposed in order to expand the functionalities of DEVSimPy allowing their enabling/disabling through a dialog window. For example, the plugin 'Blink' is proposed to visualize the activity of models during the simulation. It is based on a step by step approach and illuminates each active model with a color which depends on the executed transition function.

Building upon definitions about activity concept given in the previous section, DEVSimPy implements a new plugin called 'Activity Tracking'. This plugin increases the handling of the recent definition of the activity concept and thus opening new perspectives for the use of AT in DEVS formalism. The DEVSimPy plugin AT is generic and can be applied for any DEVS models. It does not require any modification on the DEVS simulation algorithm and does not require any additional methods in DEVS models to operate. It works in the following way:

- The user enables the plugin and chooses the set of DEVSimPy atomic models for activity-tracking.
- Before the simulation, the DEVS models are scanned in a recursive way to collect all atomic models selected by the user in the plugin interface.

- The external and internal transition functions of all selected models are decorated with a new method aimed to introduce the AT computation of these functions. A decorator function 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 composed by a key for the simulation time and an associated value for the CPU of the tracked transition function.
- From this dictionary, the Quantitative-Activity is measured by counting the number of its keys after the simulation. Moreover, knowing the code of each selected atomic DEVS model, the associated MCC can be performed before the simulation. The Weighted-Activity is performed during the simulation from the weighting transition functions defined by the modelers through the configuration of the plugin. However, concerning this point if the modeler implements the weighting transition functions, they are automatically considered for the computation of the Weighted-Activity.
- Finally when the simulation is over, the plugin offers a table resuming the QA, WA, CPU, MCC quantities for each tracked model.

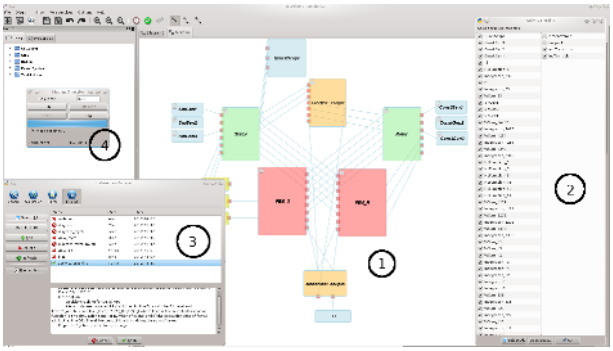

Figure 2. Activity-Tracking plugin in DEVSimPy.

Figure 2 shows the setup interface of AT plugin in DEVSimPy for a DEVS continuous model (see circle 1 in Figure 2). This model includes 49 atomic model, 15 coupled model, 203 coupling, and 3 levels of encapsulation between coupled models. It models an asynchronous electrical machine [5] employed for the diagnosis of eolian motors. The circle 2 in Figure 2 depicts the window used by the modeler to select the atomic models to be tracked (left part) and to choose the transition functions concerned with the AT plugin (right part). The circle 3 in Figure 2 depicts the window used to enabling/disabling the plugins in DEVSimPy. Having selected the set of tracked models, the simulation can be achieved (see circle 4 in Figure 2) to perform to the AT computation.

Figure 3 shows the table of results coming from the AT computation when the simulation is over. It is composed of 4 columns: the name of the model, the Quantitative-Activity, the Weighted-Activity, the CPU measure, and the MCC measure. The last row is the sum of all values of each column.

The user can obtain another form of results for AT by clicking on the name of one model. This action opens awindow plotting a graph tracing the sum of user CPU times of selected tracked functions (by default the external and internal transition functions). An example of this plotting is shown in Figure 4 for the *Integrator*_333.

4 Analyses and Discussion

In this section we first analyse the links between the four activity definitions presented in section 2. This analysis has been based from previous results obtained using the AT plugin in DEVSimPy. We

Model	QActivity	WActivity	CPU (user)	MCC
P	7007.23988904	1.99655020542	0.02	2.0
Integrator_23	48650.2655153	1.99655020542	0.119999999999	75.0
WSum_23	89892.8774337	1.99655020542	0.12	4.0
SinGen3	1001.03426986	0.900930842876	0.0	1.0
SinGen2	1001.03426986	0.900930842876	0.0	1.0
SinGen1	1001.03426986	0.900930842876	0.0	1.0
WSum_56787	454269.351664	2.0	0.64	4.0
Integrator_5678	233941.708867	2.0	0.559999999998	75.0
WSum_1234	455871.006495	2.0	0.87	4.0
Integrator_1234	235543.363699	1.98771450238	0.520000000001	75.0
WSum_2345	454269.351664	2.0	0.749999999998	4.0
Integrator_2345	233941.708867	1.99871241079	0.660000000002	75.0
NLFunction_6	123227.31862	2.0	0.329999999997	4.0
NLFunction_5	122927.008339	2.0	0.340000000001	4.0
NLFunction_3	122927.008339	2.0	0.409999999998	4.0
NLFunction_13	513530.580439	2.0	1.44	4.0
NLFunction_12	513530.580439	2.0	1.28	4.0
NLFunction_11	513530.580439	2.0	1.4	4.0
WSum_1111	197403.958017	2.0	0.379999999999	4.0
Integrator_1111	111715.424517	1.99941182173	0.240000000002	75.0
WSum_2222	201608.30195	2.0	0.330000000001	4.0
Integrator_2222	115819.665023	1.98995633726	0.259999999999	75.0
WSum_3333	196202.716893	2.0	0.37	4.0
Integrator_3333	110714.390247	1.99811460417	0.28	75.0
WSum_11	171777.480708	2.0	0.349999999999	4.0
WSum_13	171477.170427	2.0	0.27	4.0
WSum_12	171977.687562	2.0	0.29	4.0
Integrator_13	95198.3590639	1.98985571942	0.239999999999	75.0
Integrator_11	96199.3933338	1.98814166029	0.25	75.0
Integrator_12	98401.6687275	1.98247713725	0.18	75.0
Integrator3	22923.6847799	1.98310752739	0.119999999999	75.0
Integrator2	24725.5464656	1.97073081624	0.0799999999999	75.0
WSum_333	48650.2655153	2.0	0.0699999999997	4.0
Integrator_333	28729.6835451	1.98796468409	0.04	75.0
WSum_222	52654.4025948	1.99597929153	0.0599999999999	4.0
Integrator_222	32133.2000626	1.99597929153	0.0600000000004	75.0
WSum_111	49451.0929312	1.99742482158	0.0400000000004	4.0
Integrator_111	29230.20068	1.95460318929	0.0800000000004	75.0
Integrator1	23624.4087688	1.9962187835	0.0799999999999	75.0
WSum3	40241.5776485	2.0	0.0400000000009	4.0
WSum2	41442.8187723	1.97592936818	0.0699999999997	4.0
WSum1	40842.1982104	1.99742482158	0.0499999999993	4.0
Total	6405117.77572	86.3375949526	14.01	1382.0

Figure 3. Activity-Tracking results after the simulation for the machine model in DEVSimPy.

then describe the basic ideas we envision to fully exploit the activity tracking concepts developed in the previous part.

4.1 Relation Between the 4 Activity Definitions

The goal of the sub-section is to propose a discussion about the existing links between the four activity definitions: QA, WA, MCC and CPU. We have to explore how to connect together these activity notions.

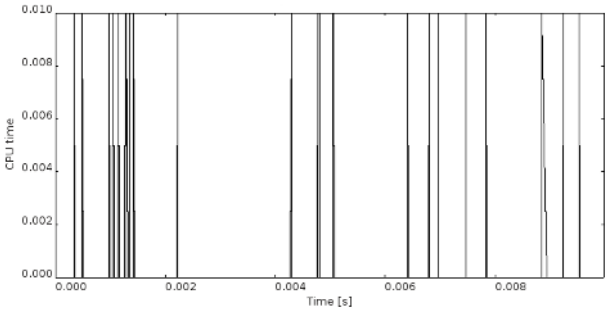

Figure 4. CPU plot for external transition of *Integrator*_333 atomic model.

First, we can point out that the QA and WA metrics are linked necessarily because of their intrinsic definitions.

Second, the link between QA and CPU is not so obvious. Usually the two metrics give the same indication: when comparing two atomic models, when the QA metric of an atomic model is greater (resp. lower) than the QA of the other one, the CPU activity is also greater (resp. lower). However, we may remark in the table of figure 3 that the atomic model *WSum*_56787 has a QA activity value equal to 454269,35 while its CPU activity is equal to 0,64. We can find an atomic model that has a QA value much lower than 454369,35 and a CPU activity which is superior to 0,64. This model is *Integrator*_2345. Therefore, is not because the QA of an atomic model is greater that the CPU time employed to execute it is greater. The sum of the CPU time of the *Integrator*_2345 transition functions is probably more important than that of *WSum*_56787.

Third, an evident link exists between MCC and CPU. Traditional, complexity measures like MCC do provide some measure of the logical structure of a program. Such measure offers the capability of describing the intra- module or procedural complexity. Even though there are no conclusive studies, this relationship is quite obvious: usually computer performance decreases as model complexity increases. However, the shape of this relationship (linear, polynomial, exponential, etc) is unknown, as are those components of complexity that mostly affect computer performance. Once again by observing the results in the table of figure 3 we can point out for example *WSum*_11 and *Integrator*2; the MCC value for *Integrator*2 is much greater than the one of *WSum*_11 and the CPU measure gives an inverse result.

For these reasons all the four different metrics propose an interesting information since they are complementary. The MCC gives information about the complexity of atomic model functions before any simulation. QA gives information about how many times the transition functions are executed while CPU gives information about the time consumption when executing the transition functions. The difference between these activity notions makes them interesting and complementary when someone has to deal with activity tracking. Instead of only relying on one kind of activity, we envision an enhanced DEVS simulation activity tracking based on QA, MCC and CPU.

In the next sub-section we describe how we plan to use these activity notions in order to be able to propose an AT-driven simulation.

4.2 An Approach Towards AT-Driven Simulation

The idea is to propose an enhanced simulation based on activity tracking concepts (called in the following AT-driven simulation). Since we have been able to define (section 2) and compute (section 3) different kinds of activity tracking notions we are able to propose an original approach for AT-driven simulation (see figure 5). When a DEVS modeler has to define a DEVS model and then

perform simulation, he has the possibility thanks to the DEVSimPy framework (specially with the AT plugin) to perform the computation of four different metrics.

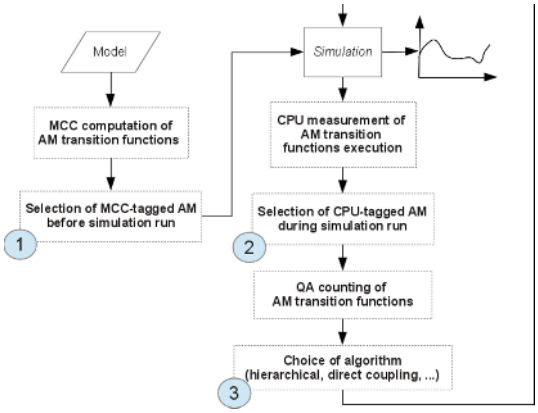

Figure 5. A proposed approach for AT-driven simulation.

To introduce activity notions in order to perform an AT-driven simulation we recommend a three part scheme as presented (see bags in figure 5).

First, the MCC measure must be used in order to perform the selection of the set of tagged atomic models (see circle 1 in figure 5) which are susceptible to consume a high amount of CPU time due to their δ_{ext} and δ_{int} functions complexity. The MCC measure gives before any simulation an estimation of potential CPU consumption of a δ_{ext} or/and δ_{int} functions.

Second, the CPU measure is computed during the simulation. As pointed in sub-section 2.3 the CPU measure gives for each tagged atomic model the CPU time consumption corresponding to the execution of their δ_{ext} and δ_{int} functions. We therefore are able during the simulation to reevaluate the atomic models that have to be tagged according to the CPU measure (see circle 2 in figure 5).

Third, the QA measure can be used during simulation in order to point out the atomic models which are the more frequently executed. A dynamic reorganization of the simulation tree can be therefore activated in order to perform a direct coupling for the atomic models having a great QA value (see circle 3 in figure 5).

We have to point out that we plan: (i) to implement an activity tracking enhanced DEVSimPy parallel simulation of DEVS models using a mutli-core machine server; (ii) to perform a dynamic switching between different kinds of simulation using the design pattern strategy during the simulation.

5 Conclusion and Perspectives

This paper deals with the links between AT concepts and DEVSimPy framework. We first introduce the activity-tracking paradigm. We describe two activity measures found in the literature (the quantitative-activity notion QA and the weighted-activity notion WA) and we introduce two new activity notions: the McCabe cyclomatic complexity (MCC) and the user time consumption measure (CPU). The implementation of these activity notions have been done in the DEVSimPy Framework by introducing a new plugin called “Activity-Tracking“. This plugin allows the user to obtain the computation of the four activity metrics (QA, WA, MCC and CPU). A complex DEVS model including 49 atomic models, 15 coupled models, 203 couplings, and 3 levels of encapsulation between coupled models points out the interest of the plugin when the user has to analyze the activity notions of a DEVS models corresponding to a real case application. We also propose an AT-driven simulation

approach which will be implemented using the DEVSimPy framework: (i) AT enhanced parallel simulation based on the MCC and CPU notions; (ii) dynamic switching between the different simulation algorithms in order to improve the CPU time consumption according to the QA activity measurement. These two kinds of enhancement are being implemented thanks to both the Strategy Design Pattern concept and the intrinsic dynamic programming feature of the Python language.

References

[1] C. H. Kung and A. Solvberg, Activity modeling and behavior modeling, in Proc. of the IFIP WG 8.1 working conference on Information systems design methodologies: improving the practice, North-Holland Publishing Co., Amsterdam, The Netherlands, 145-171 (1986).

[2] L. Liu and R. Meersman. Activity Model: A Declarative Approach for Capturing Communication Behavior in Object-Oriented Databases. in Proc. of the 18th International Conference on Very Large Data Bases (VLDB '92), San Francisco, USA, 481-493 (1992).

[3] J. S. Bolduc and H. Vangheluwe, The modelling and simulation package PythonDEVS for classical hierarchical devs, MSDL Technical Report MSDL-TR-2001-01. Montreal, Quebec, Canada: McGill University (2001).

[4] L. Capocchi, J. F. Santucci, B. Poggi, C. Nicolai, DEVSimPy: A Collaborative Python Software for Modeling and Simulation of DEVS Systems, in Proc. of 20th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Paris, France, 170-175 (2011).

[5] S. Toma, L. Capocchi, G-A. Capolino,IEEE Transactions on Industrial Electronics, **vol. PP no. 99**, 1 (2012).

[6] A. Muzy, R. Jammalamadaka, B.P. Zeigler, J.J. Nutaro, Simulation, **vol. 87 no. 5**, 449-464 (2011).

[7] X. Hu, B. P. Zeigler, Simulation, **vol. 89 no. 4**, 435-450 (2013).

[8] A. Muzy, D. Hill, What is New with the Activity World View in Modeling and Simulation? Using Activity as a Unifying Guide For Modeling and Simulation, IEEE/SMC-ACM/SIGSIM- Proc of the 2011 Winter Simulation Conference, 13 (2011).

[9] T.J. McCabe, A Complexity Measure, Software Engineering, IEEE Transactions on **vol.SE-2 no.4**, 308-320 (1976).

[10] A. Muzy, B.P., Zeigler, Activity-based Credit Assignment (ACA) in Hierarchical Simulation, IEEE/ACM/SCS: SpringSim Multi-conference, Symposium On Theory of Modeling and Simulation, Orlando, USA, Accepted for publication (2012).

[11] A. Muzy, F. Varenne, B.P. Zeigler, J. Caux, P. Coquillard, L. Touraille, D. Prunetti, P. Caillou, O. Michel, D.R.C. Hill, Simulation, **vol. 89 no. 2**, 156-177 (2013).

[12] B.P. Zeigler, H. Praehofer , T.G. Kim, *Theory of Modeling and Simulation*, (Academic Press, 2nd Edition, 2000) 510.