Algorithmic Tools for the Transformation of Petri N ets to DEVS

Article *in* Informatica · December 2013

2 authors:

Redjimi Mohamed
Université 20 août 1955-Skikda

Sofiane Boukelkoul
Université Constantine 2

Algorithmic Tools for the Transformation of Petri Nets to DEVS

Mohammed Redjimi and Sofiane Boukelkoul
Université 20 Août 1955, Faculté des sciences, Département d'informatique
21000, Skikda, Algeria
E-mail: redjimimed@yahoo.fr, Bouk.sofiane@yahoo.fr

Keywords: DEVS, Petri nets, coupling models, multi-modeling, modeling and simulation

Received: May 1, 2013

Complex systems are characterized not only by the diversity of their components, but also by the interconnections and interactions between them. For modeling such systems, we often need several formalisms and we must concern ourselves with the coexistence of heterogeneous models. This objective can be achieved by using multi-modeling. The transformation of such models in a pivot model is a technique in this context. This paper introduces the DEVS 'Discrete Event System Specification' which model coupling approach is supported by a proposal for transformation of Petri nets in DEVS models. Petri Nets are universal formalisms which offer mathematical and graphical concepts for modeling the structure and the behavior of systems. We present mechanisms which can systematically transform the places and transitions in Petri nets to DEVS models. The coupling of these models generates a DEVS coupled model capable of running on platforms based on DEVS formalism.

Povzetek: Opisana je transformacija Petri mrež v formalizem DEVS.

1 Introduction

The diversity and the complexity of increasingly growing systems has forced the scientific community to implement tools for modeling and simulation [1] [2] [3] more and more efficient and meet the expressed requirements and constraints and support the heterogeneity and especially coupling systems in various disciplines. Now, it appears essential to use federative tools which offer extensive possibilities of abstraction and formalization. The multi-modeling consists of using several formalisms when one wants to model complex systems whose components are heterogeneous [4]. The idea developed in this paper is to determine a powerful formalism and abstraction that is as universal as possible to federate a set of concepts for the expression of different models. Once the formal model described, verified and validated it comes to transforming it into an executable form. In this article, we opted for Petri nets [5] [6] as tools for formal and abstract modeling of complex systems and DEVS "Discrete Event System Specification" [7] [8] [9] as universal formalism for the coupling of several transformation models. We detail in what follows mechanisms for transforming Petri nets (PN) in DEVS models [10].It consists of an algorithm permitting to systematically transform places and transitions to atomic DEVS models.

This paper begins by introducing the concept of multi-modeling. Then, we formally define DEVS and PN specifications. The following section shows the strength of DEVS as a universal system of multi-modeling followed by a formal approach to transform PN in DEVS models. We end this paper with a conclusion and perspectives.

2 Multi-modelling

Currently, systems can achieve large degrees of complexities and heterogeneities by combining multiple aspects which requires the use of several formalisms for their representation. Multi-modeling is used to represent these systems by using different formalisms. In this case, many models based on different formalisms can coexist in a single model. According to Hans Vangheluwe [2], the paradigm of multi modeling focuses on three axes:

- Different formalisms describe the coupling and the transformation of models.
- The relationship between the models at each level of abstraction is clearly defined.
- The meta-model focuses on the description of the classes of models (models of models).

In [11] there is a representation of various possible transformations by using formalism transformation graph "FTG".

3 Related works and motivations

In multi-modeling, several researches have focused on the study of the relationship between PN or other dynamic formalism and DEVS formalisms, since DEVS is considered as one of the basic modeling formalisms based on the unifying framework of general dynamic modeling formalism. Juan de Lara and al. proposed in [12] a modeling based multi-paradigm to generate PN and State-Charts. It consists of modeling at multiple levels of abstraction implemented in AToM^3 (A Tool for Multi-formalism and Meta-Modeling) [13] [14] [15], where is presented a graphical abstraction of meta-models of Sate charts and PNs. The use of CD++ to develop PN [16] [17] is close to our work. However it only provides tools for generating PN by using library of predefining models for PN places and transitions. Therefore, one may be not finding the appropriate model for a given transition especially when it contains a big number of ports. Furthermore, in [17] we don't find a vital parallelism because firing transitions is scheduled. That means one never finds more than one transition in firing state, while the parallelism is one of the fundamental PN characteristics. Thus the conflict characteristic of PNs is silently absent, since without parallelism the problematic of conflict is not considered. So the value of our work is that is characterized by the development of algorithms that can automatically transform the existing PN in DEVS models [10]. Moreover, the most important characteristics of PNs such as parallelism, concurrency and conflict are well preserved in our approach.

4 DEVS formalism

DEVS was initially introduced by B. P. Zeigler [7] in 1976 for discrete event systems modeling. In DEVS, there are two kinds of models: atomic and coupled models. Atomic model is based on a continuous time inputs, outputs, states and functions. Coupled models are constructed by connecting several atomic models.

A DEVS atomic model is described by the following equation:

$$AtomicDEVS = (X,Y,S,\delta int,\delta ext,\delta con,\lambda, ta) \quad (1)$$

Where:

X is the set of external inputs. Y is the set of model outputs. S is the set of states. δint: $S \rightarrow S$: represents the internal transition function that changes the state of the system autonomously. It depends on the time elapsed in the current state.

δext: $S\times X\rightarrow S$: is the external transition function occurs when model receives an external event. It returns the new state of the system based on the current state. δcon: $X\rightarrow SxS$: is the transition function of conflict. It occurs if an external event happens when an internal system status changes. This feature is only present in a variant of DEVS: Parallel DEVS [8] [18]. λ: $S \rightarrow Y$: is the output function of the model. It is activated when the elapsed time in a given state is equal to its life (ta (s) represents the life of a state "s" of the system if no external event occurs).

Coupled DEVS formalism describes a system as a network of components.

$$CoupledDevs=(X_{self},Y_{self},D,\{M_d/d\in D\},EIC,EOC,IC) \quad (1)$$

Where Self: is the model itself. X_{self} is the set of inputs of the coupled model. Y_{self} is the set of outputs of the coupled model. D is the set of names associated with the components of the model, self is not in D. $\{M_d$ / d $\in$ D$\}$ is the set of components of the coupled model. EIC, EOC and IC define the coupling structure in the coupled model. EIC is the set of external input couplings. They connect the model inputs coupled to those of its own components. EOC is the external output couplings. They connect the outputs of the components to those of the coupled. IC defines internal coupling. It connects the outputs of components with entries from other components in the same coupled model.

In DEVS, both of atomic and coupled models can be represented graphically as illustrated in Fig. 1.

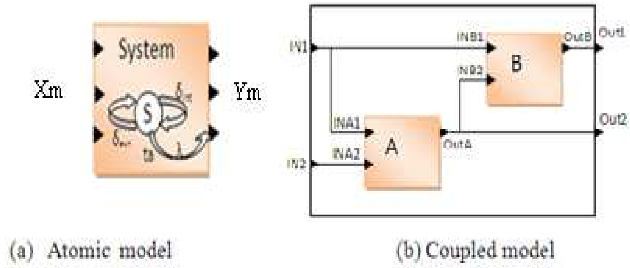

Figure 1: Representation of DEVS (a) atomic and (b) coupled models.

5 Petri nets (PN)

Petri Nets are a modeling formalism originally developed by C. A. Petri [5] [6]. They are very suitable for modeling dynamic systems.

Several types of nets can be used (timed Petri nets, colored Petri nets …) [19] [20]. We use classical Petri nets defined by the following 5-tuple:

$$PN = (P, T, PRE, POST, Mo) \quad (2)$$

P: is the set of places. T: is the set of transitions. PRE: the matrix generated by applying $P \times T \rightarrow N$. PRE [i, j] = n / n = 0 if the place is not upstream of the transition tj else n = τ / τ is the weight of the arc from pi to tj. POST: the matrix generated by applying $T \times P \rightarrow N$. POST [i, j] = n / n = 0 if the place pi is not downstream of the transition tj else n = τ / τ is the weight of the arc from tj to pi. M0: is the vector of initial marking. M[i] = k / k is the number of tokens in place pi. Fig. 2, shows a PN in the left (a) which consists of three places and one transition modeling action (T1) having two conditions (P1, P2) to be run. The result is put in place (P3).

6 PN to DEVS Transformation

6.1 Why DEVS?

DEVS provides a modular and hierarchical representation of dynamic models. Events generated by a model can take values in different areas and can be used as stimuli for other models. Also, according to B.P. Zeigler [7] [8], we can show that there is a DEVS model corresponding to each discrete event systems. We can go further, in fact, DEVS can be 'universal' [21] and allows the coupling of models and formalisms described with heterogeneous paradigms [11].

The main idea is that the models are considered as black boxes that have links with the outside world only through ports of inputs and outputs. Using this abstraction feature, several models can be coupled while enjoying the reuse of existing models. It is also possible to perform the formal verification of DEVS models, which is a valuable aid in the design of systems [22] [23].

Several DEVS-based platforms are available such as VLE (Virtual Laboratory Environment)[24][25], DEVSJAVA [26] developed in Java, Cell-DEVS (Cellular DEVS) which is based on the formalism of cellular automata [27].

The coupling of models based on DEVS is a typical task. However, non-DEVS models require an extra effort to be coupled. Two methods exist to incorporate a non-DEVS model into a DEVS environment: co-simulation and transformation [28]. The transformation of non-DEVS models (PN in our case) in DEVS models requires to specifying models in a uniform language. In the case of a co-simulation, the communications between simulators is considered. Several works such as HLA (High Level Architecture) [29] take in account this way.

6.2 Mechanisms of PN to DEVS transformation

The idea of our approach is to have as result a DEVS coupled model (CDEVS) faithful to the input PN.

6.2.1 Structure of Resulting DEVS Model

The transformation of Petri provides a DEVS coupled model where places and transitions are replaced by atomic DEVS models. Fig.3, illustrates the CDEVS model corresponding to the PN example. The DEVS model corresponding to the "transition" of PN (TDEVS for "Transition DEVS") is characterized by an output port "control" (CT1) able to send events to places upstream and verify the number of tokens or inform them about its firing. However, TDEVS receives events from the models corresponding to places upstream (PDEVS "Place DEVS") with control ports as much as number of places (CPiT1).

TDEVS is not linked by its downstream CDEVS except by output port for each AT1Pi (in black) to inform them about its crossing. All TDEVS and PDEVS are provided with an output port OutTi and OutPi (in blue). These ports are coupled directly with the output ports for eventual CDEVS output. All PDEVS have an input port (InitPi) by which they are coupled with CDEVS via an input port InitP (in green) to initialize the marking of places. The arcs from place Pi to the transition Tj are translated into output ports APiTj (PDEVS) and input ports APiTj (TDEVS) corresponding to T (black). The creation of the structure of DEVS model corresponding to the PN is performed by algorithm1 which takes as input a PN= (P, T, PRE, POST, M0). The result is a DEVS model. Algorithm1 creates links corresponding to the arcs that link places by upstream transitions thanks to PRE matrix. The POST matrix is used for the coupling between TDEVS (transitions) and PDEVS (places) downstream of the transition.

Fig. 2 illustrates the elementary transformations of PN components to their equivalent objects in DEVS. Where (a) represents a single place with the minimum of ports it has to possess. (b) Illustrates a single given transition. (c) and (d) represents the minimum of IC between a place and a transition. (e) Corresponds to a graphical representation of IC in case of conflict between two transitions. Finally (f) represents the IC of typical transformation with parallelism.

Formally, the transformation is presented as follow:

PN = (P,T,PRE,POST,Mo) $\rightarrow$
CDEVS=(X,Y,D,EIC,EOC,IC)
Where:
D = {P ∪ T}
X = {InitP, InitT}
Y = {OutDi / Di is atomic model representing Pi or Ti}
EIC = {(CDEVS.InitP, PDEVS.IntPi) ∪ (CDEVS.initT , TDEVS.IntTj)/ i ∈ N^+ & i < Number of places, j Є N^+ & j < Number of transitions }
EOC = {(Pi.OutPi, CM.OutPi) , (Tj.OutTj, CM.OutTj)/ i Є N+ & i<Number of places, jЄN^+ & j<Number of transitions }
IC = {
{(Pi.APiTj, Tj.APiTj) / PRE[i,j]>0 }
∪ {(Tj.ATjPi, Pi.ATjPi) / POST[i,j]>0 }
∪ { {Tj.CTj} X {Pi.CTjPi} / PRE[i,j]>0 }
∪ { (Pi.C PiTj , Tj.CPiTj} / PRE[i,j]>0 }
/i Є N^+ & I < Number of places, j Є N^+ & j < Number of transitions

}

Algorithm 1 : Transformation PN To DEVS

```
Main_PN_DEVS

Input PN= (P,T,PRE,POST,M0)
Output CDEVS //coupled model
Begin :
Create CDEVS as coupled DEVS model //void model
For all transition i do
   create TDEVSi as atomic DEVS model
end for
for all places j do
   create PDEVSj as atomic DEVS model
end for
for all PDEVSj do
    add 'InitPj' as intput port and join it to
   CDEVS.IN.InitP  //starting tokens
   add 'OutPj' as output port and join it to
   CDEVS.OUT.OutPj  //output stream
end for
for all TDEVSi do
   add 'InitTi' as input port //initialize, stop, pause,
   release
  join 'InitTi' port to CDEVS.IN. InitT port //coupling
  add 'OutTi' as output port and join it to
  CDEVS.OUT.OutTi  //output stream
  add 'CTi' as output port // control: check, reserve,
  decrement, cancel
```

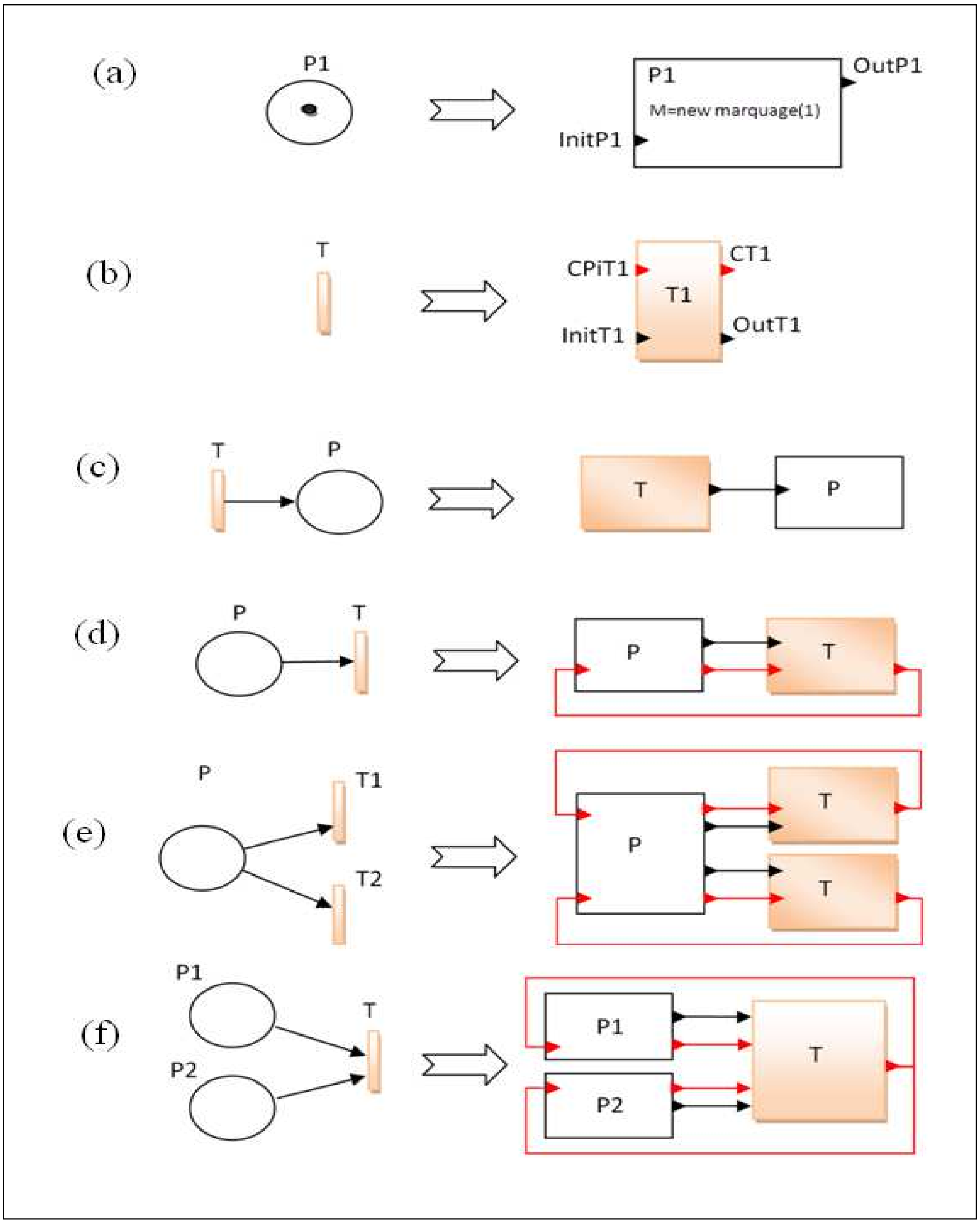

Figure 2**:** Graphical representation of elementary transformations and IC between generated DEVS models

for all PDEVSj **do**

if (PRE[i,j] > 0) //upstream place
add to PDEVSj ‘CTiPj’ as input port //check, reserve, decrement, cancel
join TDEVSi.OUT.CTi to PDEVSj.IN.CTiPj // coupling
add to PDEVSj ‘CPjTi’ as output port //ok, busy ,number_of_free_tokens
add to TDEVSi ‘CPjTi’ as input port //ok, busy ,number_of_free_tokens
join PDEVSj.OUT.CPjTi to TDEVSi.IN. CPjTi // coupling
add to PDEVSj ‘APjTi’ as output port //arc: value = PRE[i,j]
add to TDEVSi ‘APjTi’ as intput port //arc: value = PRE[i,j]
join PDEVSj.OUT. APjTi to TDEVSi.IN.APjTi // coupling
end if
if (POST[i,j] > 0) //downstream places
add to TDEVSi ‘ATiPj’ as output port //arc: value = POST[i,j]
add to PDEVSj ‘ATiPj’ as input port //arc: value = POST[i,j]
join TDEVSi.OUT.ATiPj to PDEVSj.IN.ATiPj // coupling
end if
end for
end for
end Main_PN_DEVS

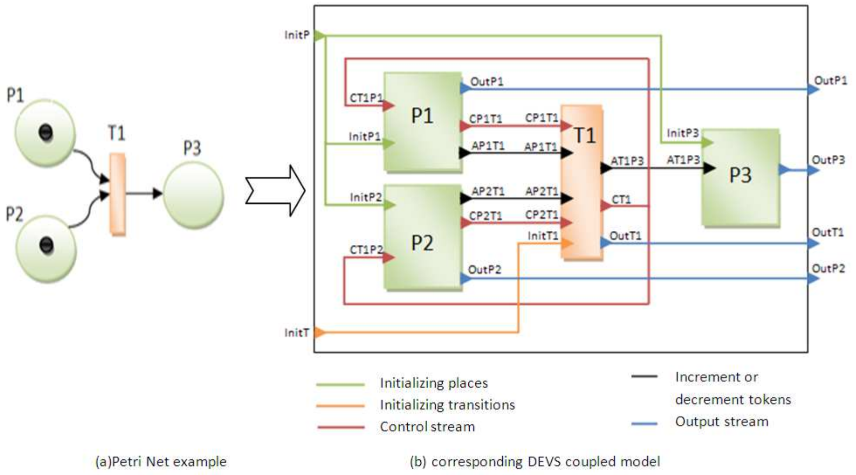

Figure 3**:** PN to coupled DEVS transformation.

6.2.2 Dynamic of Resulting DEVS Model

The dynamic of generated DEVS model is controlled by the functions of DEVS formalism which are δint, δext and λ. After initialization of places (PDEVS) by the initial marking and after launching the evolution of the model by the event "initialize" received by all transitions (TDEVS), the latter are in state "checking" (by δext) to see if the number of tokens in places upstream is sufficient to achieve a crossing. Event "check" is sent by λ. After receiving the event, PDEVS transmit the number of their free tokens (which are not reserved by other transition) with λ as well. If the number of tokens is sufficient to validate the transition (TDEVS), the status is changing from "checking" to "reserving" and the event "reserve" is sent with λ. The firing does not occur directly. It must go through a reservation status to avoid conflicts (if places are upstream of several transitions), as long as the transitions are in continuous competition. In this way the properties of PN in terms of dynamics and competition is faithfully preserved in our transformation approach.

When PDEVS receives the event "reserve" it returns "ok" if there is still enough free tokens, otherwise, it returns "fail". If TDEVS receives at least one "fail", it returns immediately the signal "cancel" to release the reserved tokens. It puts its state "Validated" otherwise. At this point, the transition can pass the crossing and therefore returns "decrement" to PDEVS which will destroy the tokens reserved by TDEVS in question. It sends simultaneously "increment" to PDEVS located downstream in order to increment the number of tokens with the value received by the input port (weight of arc). After firing a TDEVS, it rehabilitates "checking" and so on.

Functions δ_{ext}, δ_{int}, δ_{con} and λ, characterizing the models TDEVS, are summarized in Table2. The first two columns represent the inputs, which are the events and the current state. The other columns show the outputs of each function. The table rows are grouped separately for each current state and models PDEVS. Functions are shown in Table 2. By convention, if all events have the same impact, we write "all events". Empty cells indicate the absence of values, for λ that means the absence of events and for δ_{ext}, δ_{int} and δ_{con} that the function does not produce an output state. The "&" symbol indicates that the events are simultaneous.

Event	**Current state**	δ_{ext}	δ_{int}	δ_{con}	**λ (current state)**
initialize	all states	checking			Out
pause		Paused			Out
Stop		Stopped			Out
release		checking			Out
Free tokens	Reserving		reserving	Reserving	reserve
Ok		validated, reserving		validated, reserving	
fail		canceling		Canceling	
all events	Validated		checking		decrement & increment & out
all events	Canceling		checking		cancel

Table 1: The outputs of the TDEVS model functions.

6.2.3 Example of Transformation

Fig. 4 and 5 present an example of transformation of one of famous case study in PN training field: Producer-Consumer (Prod_Cons_PN).

The formal definition of this PN is:

Prod_Cons_PN = (P, T, PRE, POST, M0)

P = {P1, P2, P3, P4, P5, P6}

T = {T1, T2, T3, T4}

$$PRE = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} POST= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} M_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 7 \end{pmatrix}$$

P1 (Producer is ready to produce), T1 (Begin of production), P2 (Production is run), T2 (End of production), P3 (plug containing products, initially, plug is empty), P4 (Consumer is ready to consume), T3 (Begin of consummation), P5 (Consummation is run), T4 (End of consummation) and P6 (Number of free puts, initially: all puts in plug are free).

Event	Current state	δ_{ext}	δ_{int}	δ_{con}	λ (current state)
initialize	all states	Checking		Checking	Out
check	checking	Checking	Checking	Checking	free_ tokens
reserve		Reserving		Reserving	
increment		Incrementing		Incrementing	
decrement		Decrementing		Decrementing	
cancel		Checking		Checking	
check	reserving	Reserving	Checking	Reserving	ok, fail
reserve		Reserving		Reserving	
increment		Incrementing		Incrementing	
decrement		Decrementing		Decrementing	
cancel		Checking		Checking	
check	incrementing	Checking	Checking	Checking	Out
reserve		Reserving		Reserving	
increment		Incrementing		Incrementing	
decrement		Decrementing		Decrementing	
cancel		Incrementing		Incrementing	
check	decrementing	Checking	Checking	Checking	Out

Table 2: The outputs of the PDEVS model functions.

Fig.4 represents the coupled model faithful to the PN modeling Producer-Consumer. Fig.5 illustrates the corresponding coupled DEVS model. We conserve the same color signification as shown in Fig. 3: Color green to initialize places' tokens number. Color orange to initialize transitions. Color red: to illustrate control stream. Color black: to illustrate tokens incrementing or decrementing and color blue for outputs.

6.2.4 Discussion

Petri nets are formal tools modeling dynamic systems dealing perfectly with the aspect of competition, concurrency and parallelism. Therefore, they require gentle handling during mapping in order to not lose their specifications. In our approach, competition is preserved by the creation of temporary state transitions which is the reserving state. Thus, a token cannot participate at the same time, in the firing of two transitions in conflict. However, the transition must immediately release tokens if it fails to be validated in order to not paralyze other transitions which are in conflict with it.

In this paper we presented the generalized PN for the reader to understand the mechanism of transformation. However, other extensions such as coloured PN can also be processed. In this case, tokens will no longer be trivialized. We will need to extend the type of representation to comprise a list with different colours. Thus, during the broadcast of the event "check" with a transition. Places of upstream should check the port connecting to the transition in order to send only the number of free tokens with the same colour as specified

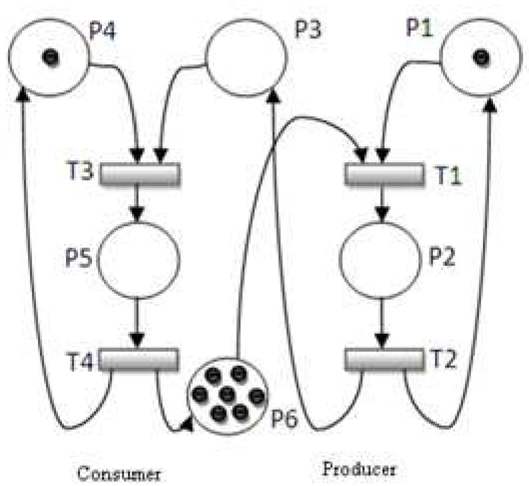

Figure 4: PN Producer-Consumer.

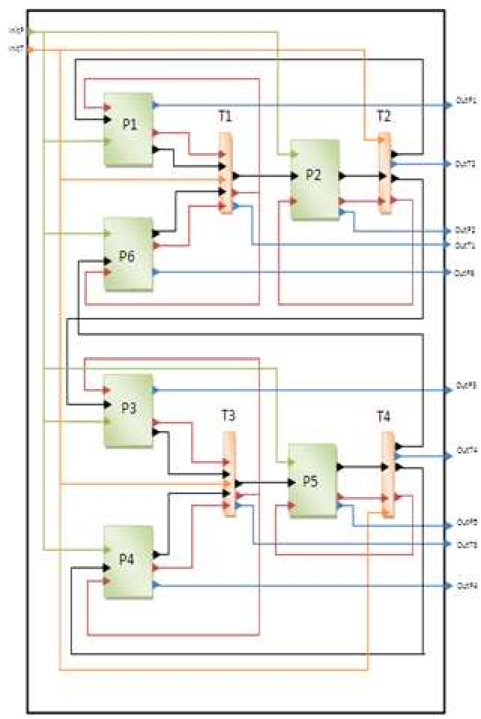

Figure 5: DEVS coupled models corresponding to

at this port.

In addition, the DEVS formalism provides flexibility in the internal structure of the models [30]. Models may disappear, others can take over. This aspect of dynamic structure related to DEVS will simplify the complexity of PN related to the representation of structural changes in systems. Therefore one DEVS model can represent several PNs at a time.

7 Conclusion and perspectives

In this paper we have presented a transformation approach of Petri nets to DEVS models, where places and transitions are transformed to atomic models. Coupling these models generates a coupled DEVS. This work falls within the framework of multi-modeling and transformation models based on multi-formalisms. Our choice of DEVS as focal formalism was based on its power in unifying and coupling models. Characterized by its abstraction, implementations independence and its ability to model complex systems in the form of a hierarchical model, DEVS is a formalism that can be the unifier of models.

By the transformation presented in this paper, the PN can enjoy the simulation on multiple DEVS based platforms.

Our perspectives focus on the implementation of such transformations to modelling complex industrial systems such as petroleum plants.

References

[1] Fishwick, P.A. (1995). Simulation Model Design and Execution: Building Digital Worlds. *Prentice Hall: Englewood Cliffs*, NJ.

[2] Vangheluwe, H. (2008). Foundations of modelling and simulation of complex systems. Electronic *Communications of the EASST, 10: Graph Transformation and Visual Modeling Techniques* .http://eceasst.cs.tuBerlin.de/index.php/eceasst/issue /view/19.

[3] Pidd, M. (2004). Systems Modelling: Theory and Practice. *John Wiley & Sons: Hoboken*, NJ.

[4] Fishwick, P.A. (2004). Toward an integrative multimodeling interface: A human-computer interface approach to interrelating model structures. *Simulation* **80**(9): 421.

[5] Murata, T. (1989). Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, Vol.77,No.4 pp.541-580, April 1989.

[6] Peterson, J.L. (1977) : Petri nets, *Computing Surveys*. pp. 223–252

[7] Zeigler, B. P. (1976) : Theory of Modelling and Simulation, *Wiley InterScience*.

[8] Zeigler, B. P. Praehofer, H. and Kim, T. G.(2000): Theory of Modeling and Simulation, *Second edition. Academic Press, ISBN 0127784551*

[9] Shafagh, J. and Wainer, G.A. (2011). A Performance Evaluation of the Conservative DEVS Protocol in Parallel Simulation of DEVS-based Models., *Proceedings of 2011 Spring Simulation Conference (SpringSim11), DEVS Symposium*, page 103--110 - April 2011

[10] Boukelkoul, S. and Redjimi, M. (2013). Mapping Between Petri Nets and DEVS Models. *Proceeding of the 3rd International Conference on Information Technology & e-Service*, Sousse, Tunisia

[11] Vangheluwe, H. (2000). DEVS as a common denominator for multi-formalism hybrid systems modeling. *Conference IEEE International Symposium on Computer-Aided Control System Design*, Alaska, pp.129-134.

[12] De Lara, J. and Vangheluwe, H. (2002). Computer Aided Multi-Paradigm Modeling to Process Petri-Nets and Statecharts. *Lecture Notes in Computer Science, Springer* Volume 2505, pp 239-253.

[13] Home page: http://atom3.cs.mcgill.ca/ De Lara, J.,

[14] De Lara, J and Vangheluwe H. (2004). Meta-Modelling and Graph Grammars for Multi-Paradigm Modelling in AToM3. Manuel Alfonseca. Software and Systems Modeling, Vol 3(3), pp.: 194-209. *Springer-Verlag. Special Section on Graph Transformations and Visual Modeling Techniques.*

[15] De Lara, J., Vangheluwe, H. (2005): Model-Based Development: Meta- Modelling, Transformation and Verification, *The Idea Group Inc*, pp. 17 (2005).

[16] Wainer, G.A. and Mosterman, P. (2011) Discrete-Event Modeling and Simulation: Theory and Applications. *Taylor and Francis*.

[17] Jacques, C. J. D. and Wainer, G. A. (2002). Using the CD++ DEVS Tookit to Develop Petri Nets. *Proceedings of the SCS Summer Computer Simulation Conference,* San Diego, CA. U.S.A

[18] Shafagh, J. and Wainer, G.A.(2011). Conservative Synchronization Methods for Parallel DEVS and Cell-DEVS. *Proceedings of the 2011 ACM/SCS Summer Computer Simulation Conference*, The Hague, Netherlands.

[19] Genrich, H. J. and Lautenbach, K. (1981) System Modelling with High-Level Petri Nets. *Theoretical Computer Science*, vol. 13 (1981)

[20] Jensen, K. and Kristensen, L.M. (2009) Coloured Petri Nets Modelling and Validation of Concurrent Systems. *Springer*.

[21] Touraille, L., Traoré, M. K. and Hill, D. R. C. (2010). SimStudio: une Infrastructure pour la modélisation, la simulation et l'analyse de systèmes dynamiques complexes. *Research Report LIMOS/RR*-pp.10-13.

[22] Byun, J.H. Choi, C.B. and Kim, T.G. (2009) Verification of the devs model implementation using aspect embedded devs. *In Proceedings of the 2009 Spring Simulation Multiconference*, San Diego, USA, 2009

[23] Freigassner, R. Praehofer H. and Zeigler, B. P.(2000). Systems approach to validation of simulation models. *Cybernetics and Systems*, pp.52–57.

[24] Quesnel G. Duboz, R. and Ramat, E. (2009). The Virtual Laboratory Environment – An operational

framework for multi-modeling, simulation and analysis of complex dynamical systems. *Simulation Modeling Practice and Theory*, 17 :641–653.

[25] Quesnel, G. (2006). Approche formelle et opérationnelle de la multi-modélisation et de la simulation des systèmes complexes. PHD trésis Laboratoire d'Informatique du Littoral (LIL). Calais - France

[26] Sarjoughian, H. and Zeigler, B. P. (1998). Devsjava: Basis for a DEVS-based collaborative ms environment. *SCS International Conference on Web-Based Modeling and Simulation*, San Diego, CA, vol. 5, pp. 29-36.

[27] Ilachinski, A. (2001). Cellular Automata, a Discrete Universe, *World Scientific Publishing Co*, ISBN 981-02-4623-4.

[28] Schmidt, D. C (2006) .: Model-Driven Engineering *Guest Editor's Introduction IEEE Computer*, Vol. 39, No. 2, pp. 25-31.

[29] IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Framework and Rules, Institute of Electrical and Electronics Engineers, IEEE (2000) 1516-2000

[30] Baati, L. (2007) : Approche de modélisation DEVS à structure hiérarchique et dynamique. *LSIS UMR-CNRS 6168, Domaine Universitaire de St Jérôme*.