DOI: https://doi.org/10.31341/jios.44.2.4 UDC 004.414.23:005.4
Open Access Preliminary Communication

A DSDEVS-Based Model for Verifying Structural Constraints in Dynamic Business Processes

Sofiane Boukelkoul *sofiane.boukelkoul@univ-constantine2.dz*
LIRE Laboratory
University of Abdelhamid Mehri – Constantine 2, Algeria

Ramdane Maamri *ramdane.maamri@univ-constantine2.dz*
LIRE Laboratory
University of Abdelhamid Mehri – Constantine 2, Algeria

Abstract

This paper presents a DSDEVS-based model "Dynamic Structure Discrete Event System specification" for modeling and simulating business processes with dynamic structure regarding to different contexts. Consequently, this model, formally, improves the reuse of configurable business processes. Thus, the proposed model allows the analysts to personalize their configurable business processes in a sound manner by verifying a set of structure properties, such as, the lack of synchronization and the deadlock by means of simulation. The implementation was done in DEVS-Suite simulator, which is based on DEVSJAVA models.

Keywords: Business Process, DEVS, DSDEVS, Dynamic structure, Configurable Business Processes

1. Introduction

Configurable business process models are designed in a generic way to gather similar functionalities of different processes in one common model. This model is configured in order to result a tailored variant according to the analyst requirements [1]. However, traditional configuration approaches may result model variants without preserving soundness. Moreover, traditional simulation methodologies provide support only for changes in model descriptive variables [2]. Many dynamic systems are better represented by models with both changes in state variables and changes in their structure. The DSDEVS formalism "Dynamic Structure Discrete Even System Specification" is well suitable for this purpose. This formalism was introduced in [2], [3] and provides full support for systems with dynamic structure. Related works in the field of modeling and simulation of dynamic structure systems can be found in [3], [4], [5], [6] and [7].

In this paper, we are interested in business processes "BP" with different contexts as dynamic structure systems. These processes are often called configurable

processes. They contain common elements, which are present in all configurations, and configurable elements, which can be personalized regarding to the business functional purpose. Commonly, in such processes, the task of configuration is confided to the analysts. However, they are not under the lee of human mistakes, and the resulted variants may not have sound in their structure because of the non-respect of the structural constraints. To cope with this aim, we provide a DSDEVS-based model to verify structural constraints at both design and execution time of the configurable business processes by means of simulation. For this purpose, we use the DEVS-Suite simulator [8].

The reminder of this paper is organized as follows. In section 2, we present the related works and the motivations. In section 3, give some definitions as a background. In section 4, we formally, define the proposed model. Then we illustrate the implementation with the results in section 5. Finally, we conclude and highlight some overviews in section 6.

2. Related Works

In business process modeling fields, the configurable models are getting attention as flexible approaches for business process design by reuse [1]. Such models are designed in a general way in order to group common and variable parts of similar processes. At design time, these models are configured by personalizing the configurable elements. We call the resulted processes variants. Several approaches have been proposed for the configuration of the configurable business processes [9], [10]. Other contributions such as in [11], [12] come as guide tools to assist the analysts in the configuration. However, many configurations result business process variants without soundness in their structure because of the human faults committed by the analysts at design time when choosing options on the configurable elements inside the business processes.

This paper proposes a DSDEVS model in the purpose to ensure soundness of the configured business process variants at design time. For this aim, many researches are done, such as, [11] and [13], where authors define the requirements for a correctness preserving configurable process modeling based on the behavioral correctness of the expected process variant. In [14], authors proposed an Event-B approach to check the configuration procedure correctness of business processes modeled in BPMN "Business Process Model and Notation" [15] . In addition, the use of Petri nets for the properties verification in workflow specifications [16], [17]. In [18], the Provop framework "PROcess Variants by OPtions" [19] [20] is used to guarantee the soundness process variant. However, these approaches are, mainly, focused on the process configuration at design time only and do not offer a practical use for the analysts.

To cope with these aims, we provide a DSDEVS model to verify structural properties in business processes at both design and execution time.

3. The DSDEVS formalism

DSDEVS for "Dynamic Structure Discrete Event System Specification" is a formalism for specifying systems where the structure is not static. It was defined by F. Barros [2] as an extension of DEVS "Discrete Event System Specification" which was, initially, introduced by B. P. Zeigler [21] in 1976 for discrete event systems modeling. The DSDEVS has been used to model and simulate systems with dynamic structure such as in [7], [22], [23] and [24]. In this paper we use the DEVS formalism, extended by DSDEVS in order to model and verify business processes with dynamic structure.

This section presents the DSDEVS formalism according to F. Barros [2]. In this formalism there are two kinds of models: basic models and network models.

3.1. Basic models

Basic models are defined as in DEVS formalism, based on a continuous time, inputs, outputs, states and functions. Formally, DSDEVS basic models are described by the following equation:

$$M = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta)$$

Where:

- X is the set of external inputs (ports).
- Y is the set of model outputs (ports).
- S is the set of model states.
- δ_{int}: $S \rightarrow S$: represents the internal transition function that changes the state of the system autonomously. It depends on the time elapsed in the current state.
- δ_{ext}: $S \times X \rightarrow S$: 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.
- 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.

3.2. Network models

Network models are composed of DSDEVS basic models. Contrarily, in other simulation formalisms, that handle static structure models only, the structure of the DSDEVS network can be changed during the simulation. Formally, the DSDEVS dynamic structure network "DSDEVN" is described by the following equation:

$$DSDEVN = (X , M_x)$$

Where:

- X: is the DSDEVS network executive.
- M_x: model of x.

The DSDEVS network is defined with these two components: the network executive and the model of the executive, which is a DSDEVS basic model and is formally defined by the structure:

$$M = (X(x), Y(x), S(x), \delta_{int}(x), \delta_{ext}(x), \lambda(x), ta(x))$$

All the information about the dynamic structure network, such as the composition and the coupling, is located in the state of the executive "X".

The state sx ∈ S(x) contains information about the structure of the DSDEVS network. It is formally defined by:

$$M = (X_{\Delta}, Y_{\Delta}, D, \{M_i\}, \{I_i\}, \{Z_{ij}\}, SELECT, \theta) .$$

Where:

- X_{Δ}: is the input event set of the DSDEVS network, with Δ is a DSDEVS dynamic structure network.
- Y_{Δ}: is the output event set of the DSDEVS network.
- D: is the set of the components.
- M_i: is the model of the component I, for all i∈D.
- I_i: is the influence of I, for all i∈D ∪ {X, Δ}.
- Z_{ij}: is the translation function from i to j, for all j∈I_i.
- SELECT: is the select function.
- θ:is the state of other variables not defined before.

The state variables are subject to the following constraints:

- X ∉ D
- M_i is a basic model and is defined by the structure: $M = (X_i, Y_i, S_i, \delta_{int}\, i, \delta_{ext}\, i, \lambda_i, ta_i)$, for all i∈ D, i∉ Ii , for all i∈ D ∪ {X, Δ}.
- SELECT: Π→D ∪ {X}, where D ∪ {X, Δ} where: Π = 2 D ∪{X} – {} and SELECT (A) ∈ A.
- $Z_{\Delta j}: X_{\Delta} \rightarrow X_{j`}$
- $Z_{i\Delta}: Y_i \rightarrow Y_{\Delta`}$
- $Z_{ij}: Y_i \rightarrow Y_{j`}$
- $Z_{kx}(y) \neq Ø \Rightarrow Z_{k\,j}(y) = Ø$, for k ∈ D ∪ { Δ } and for all j ∈ I_k– {X},

with Ø ⇔ null event.

This constraint guarantees that if a model sends an external event to the executive, then the executive is the only element that receives the event. By this way, we prevent the ambiguity when the executive and other several components receive an event.

4. The proposed model

4.1. Overview of the model architecture

Figure 1 illustrates the architecture of the overall model as a system, where the inputs are the business process in BPMN and a set of structural configuration. Then the BP goes through the transformation to the DSDEVS formalism by applying the rules presented in [25] and extended in this paper. The resulted model consists of a variant of the initial configurable business process implemented in DEVSJAVA. This variant will undergo a simulation plan in DEVS-Suite simulator v4.0 [26] to improve the concordance and the soundness regarding to the structural side. In addition to the setting of configurable business processes known in the literature, our model allows as well, the dynamic self-changes applied at the time of simulation. This feature is guaranteed by the DSDEVS formalism by exchanging structural events between the internal components of the BP variant. Thus by means of simulation, the modeler will analyze the resulted outputs. If he detects errors in the execution termination explaining the less soundness in the structure of the model then he will have another opportunity to reconfigure the initial business process by applying another set of structural constraints. Then the process should restart again. The simulation outputs tell, as well, if the execution runs in concordance with the structural constraints.

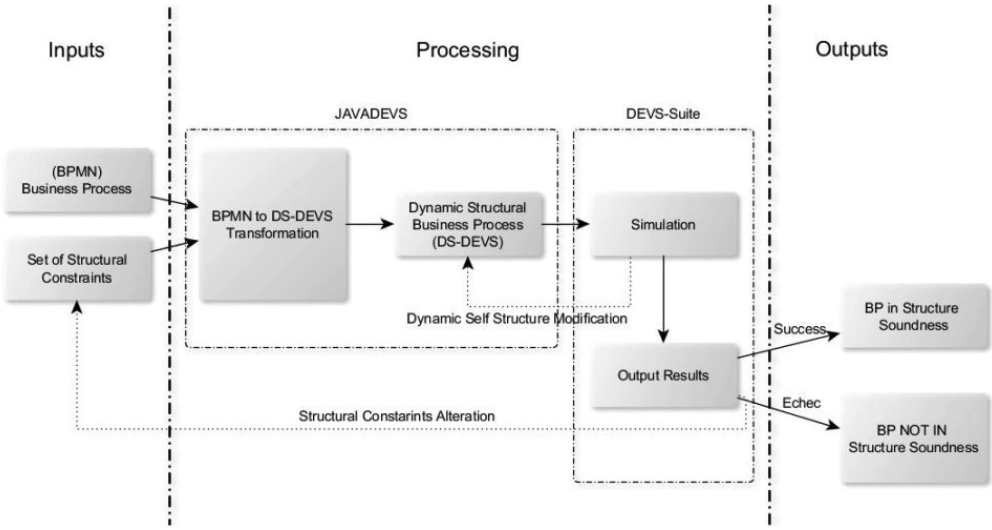


Figure 1. Architecture of the overall model.

In this approach, we extend the meta-model of DEVS presented in [25] in order to handle configurable elements of C-BPMN "Configurable Business Process Modeling Notation" [27] such as configurable gateways which are presented in Table.1.

From-To	OR	XOR	AND	Sequence
OR	*	*	*	*
XOR		*		*
AND			*	

Table 1. C-BPMN gateways constraints [28].

The reader can refer to [25] for more detail about the transformation of BPMN models to DEVS formalism.

The configuration of the gateway OR can result one of these element OR, XOR, AND or Sequence, while the XOR can be configured only to XOR or Sequence. However, a configurable AND can only be mapped to an AND.

Formally, the atomic DEVS model which represents the activity in the business process presented in [25] will be redefined as the following:

$$DEVS = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta)$$

$X = \{In\}$, $Y = \{Out,\ Out\_DSDEVS\}$, $S = \{S_0, S_1, S_{DSDEVS}\}$
$\delta_{ext}(\{e_1,e_2\},S_0)=S1$, $\delta_{int}(S_1)=S_0$,
$\lambda(S_1)=\{e_3\}$,
$\lambda(SD_{SDEVS}) = \{addModel, removeModel, replace, link, unlink, removelinks, stop\}$
$ta(S_0)=\infty$, $ta(S_1)=0$, $ta(SD_{SDEVS})=0$

The detail of the dynamic applied on the structure is explained in the implementation section.

The meta-model of DSDEVS should include a sub model called (The executive) which has one input port. This port is used to make changes on the network structure, depending on the received events. In this way, models or links can be created or removed during the simulation. The simulation ends when the executive receives the message "stop" through the input port. The external transition function of the executive DSDEVSx model is responsible of the changes applied on the whole structure by receiving messages through the in port In_DSDEVS as illustrated in figure 5, such as:

addModel: aModel, adds a new model to the simulation.

removeModel: aModel, removes a model from simulation. All links from and to the model are removed.

replace: aModel with: bModel, replaces a model by a new one. Links remain the same.

link: aModeport: aPortto: bModelport: bPort, creates a link between two models.

unlink: aModel port: aport to: bModel port: bPort, deletes a link between two models.

removelinks: aModel port: aport, removes all links

find: aName, find the model with a aName;

In the following, we formally define the main function in the model, which is the external transition function of the executive model. It is responsible of the structural changes in the BP model:

External Transition Function of the executive model:

```
external: aMode elapsed: e port: aPort id: i value: value
aPort:= #In_DSDEVS if True: [ "Message received on the
In_DSDEVS port"
    action := value at: 1. "Obtain the event action"
```

"Add a model"

```
    action = 'addModel' isTrue[
    targetModel := value at: 2. "Obtain the name of the
    model to be added"
    links := value at: 3 "Obtein the list of links"
    model := self model: targetModel. "Create themodel"
    selfaddModel: model. "Add the new model to the network"
    self link: model port: #Out_DSDEVS to: self port:
    #In_DSDEVS. "Link model port #Out_DSDEVS"

          [i<liks size] whileTrue: [
          aLink := Links at: i.
          fromModel := aLink at:1.
          fromPort := aLink at:2.

          toModel := aLink at:3.

          toPort := aLink at:4.
              super link: fromModel port: #fromPort to:
              toModel port: #toPort. "Link model port #Out"
              i:=i+1
              ]

    ]
```

"Remove a model"

```
    action = 'removeModel' isTrue[
    targetModel := value at: 2. "Obtain the name of the
    model to be removed"
    selfremoveModel: targetModel."Remove the targetModeland
    its links"
          ]
```

"Create new connections"

```
    action = 'addConnection' isTrue[
    [i<liks size] whileTrue: [
          aLink := Links at: i.
          fromModel := aLink at:1.
          fromPort := aLink at:2.

          toModel := aLink at:3.
          toPort := aLink at:4.
              super link: fromModel port: #fromPort to:
              toModel port: #toPort.
```

```
                i:=i+1
                ]

    ]
```

"Drop connections"

```
    action = 'dropConnection' isTrue[
       [i<liks size] whileTrue: [
         aLink := Links at: i.
         fromModel := aLink at:1.
         fromPort := aLink at:2.

         toModel := aLink at:3.
         toPort := aLink at:4.

             superunlink: fromModel port: #fromPort to:
             toModel port: #toPort.
             i:=i+1
       ]

    ]
].
^self continue: e
```

4.2. Case Study

The figure 2 illustrates a configurable process model of travel booking process. This process includes five main functionalities: (1) flight searching (i.e. activity a1), (2) flight booking with alternatives, (3) recommendation (ending at a12), (4) discount offer (activity a13) and (5) payment (till activity a16). The process is modeled using the C-BPMN. We consider four main control flow elements: activity (represented with a rectangle), edge (control flow edges represented with arrows), event (represented with a circle) and connector (represented with a diamond). Thereby, an activity is the main element of a process model and describes the work to be done. The control flow edges describe precedence dependency between nodes (i.e. activities and connectors). An event is something that happens during the execution of a business process. The connectors model parallel, split and alternative executions. This example includes 7 configurable activities, which are highlighted with a thicker border in the model. The other activities are non-configurable, so they should be included in every configured variant. Whereas, the activity a10 or a11, for example, may vary from one process to another, as they are configurable. A gateway may be configured to restrict its behavior. It can be configured by changing its type (e.g. from OR to AND), or/and restricting its incoming or outgoing branches. Configurable gateways are configured according to a set of configuration constraints [28]. In Table.1 each row corresponds to a configurable gateway that can be configured to one or more of the gateways in columns. For example, a C-BPMN configurable OR can be configured to any connector's type while a C-BPMN configurable AND can only be configured to a BPMN AND. In C-BPMN, the gateway AND should never be configured to a sequence, since C-BPMN gateways can be configured to restrict their behavior.

We note that C-BPMN does not change the rules of BPMN syntax, since objects in C-BPMN are extended from BPMN ones and changes are only applied on the attributes but not the entities. This guarantees that C-BPMN configured objects behave as the BPMN ones they are mapped to.

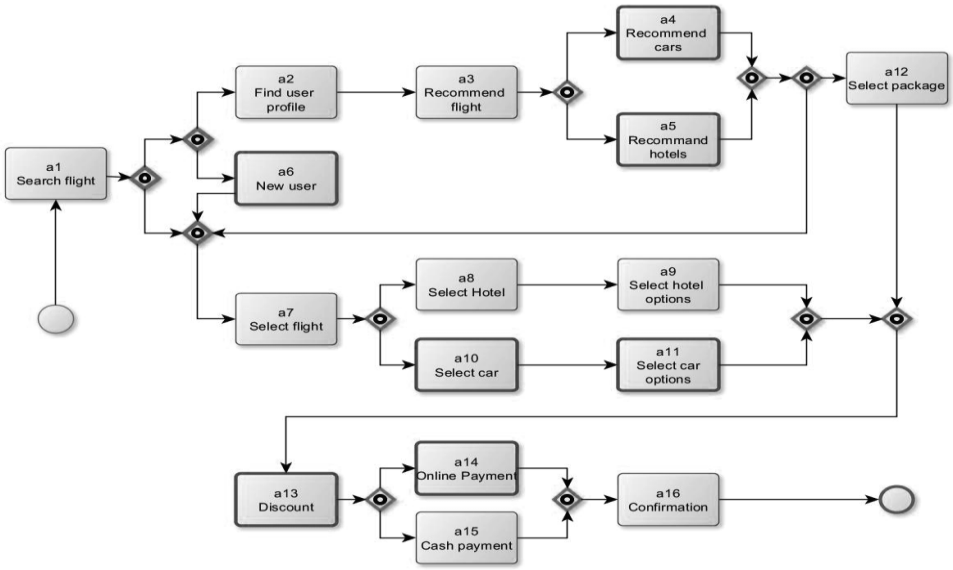


Figure 2. C-BPMN Booking BP with Dynamic Structure (Inspired from [29]).

Figure 3 illustrates the DSDEVS graphical representation of the booking BP, Initially, presented in figure 2. However, in figure 4 we show an example of a process variant derived from the configurable process in figure 2. In this variant the process analyst does need neither the recommendation functionality for cars (activity a4) nor the option to rent a car (activities a10 and a11). This refers respectively to the sequence configuration starting from a3 to a12 by removing both of activities a4 and a5. This configuration implies fictionally, to remove a10 and a11 also. Another configuration was applied on the payment step consisting of the exclusive choice of the payment type (online or cash payment). The activity of the discount option is configurable and it is removed in this process variant.

In C-BPMN, only configurable elements can be altered. However in DSDEVS all sub-models (activities) can be configured, added, dropped or moved from one place to another.

The executive model uses the In_DSDEVS in port to receive messages from either the sub models (which are linked by dotted lines, graphically, representing the internal couplings between the components and the executive model) during the simulation or from the outside by manually pre-inserting series of messages (events) by the analyst in order to configure the BP model regarding to a functional context.

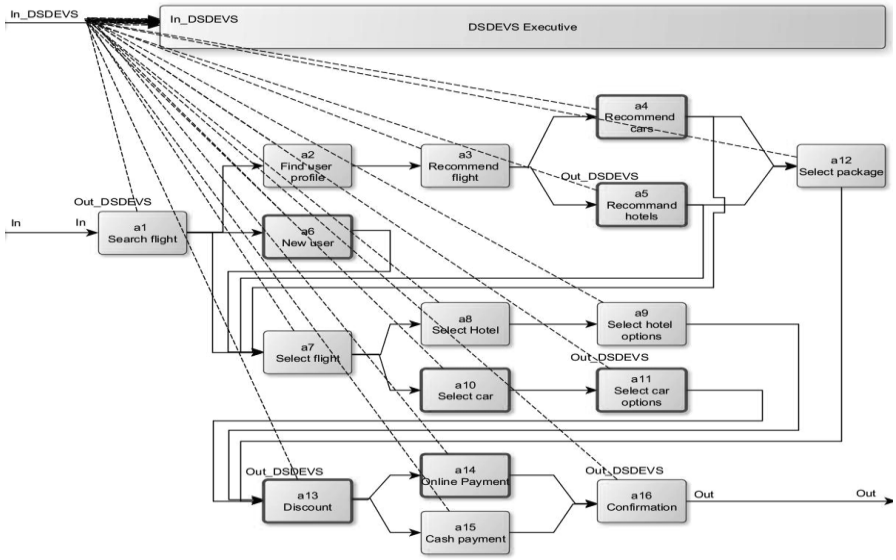


Figure 3. The general graphical representation of the Booking BP in DSDEVS.

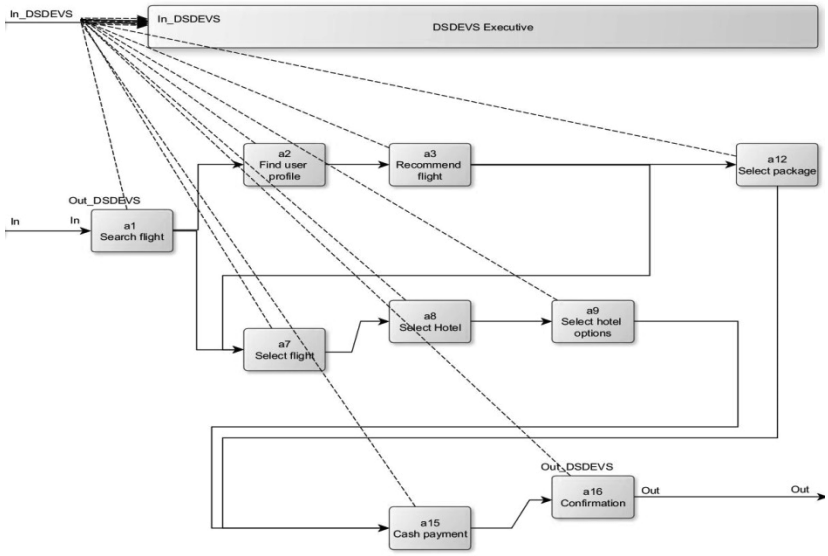


Figure 4. Graphical example of a variant of Booking BP in DSDEVS related to a specific context.

5. Simulation and Results

The implementation was done in DEVS-Suite simulator v 4.0 [26], which is based on DEVSJAVA [8] models. Firstly, models are defined in java in accordance with DEVSJAVA models. Then compiled models ".java" result java classes ".class",

which will be simulated in DEVS-Suite. The latter simulator provides logs and analysis tracking tools.

Contrarily to traditional approaches of business process configuration that apply the configuration manually and before execution, this approach allows the analysts to make both of manual personalization before the execution and make structural changes when the model variant is running. The first option can be reached by either manual configuration directly on the source model or by injecting a series of messages to the executive model. The second option is ensured by sending messages (events) from activity models to the executive one inside the business process during the execution. In this section, we illustrate the simulation result of a variant of the booking business process.

Figure 5 illustrates the presentation of the booking business process in DEVS-Suite simulator. After applying a proposed configuration according to the specific needs of figure 3, the variant of the initial business process will be generated (see figure 6).

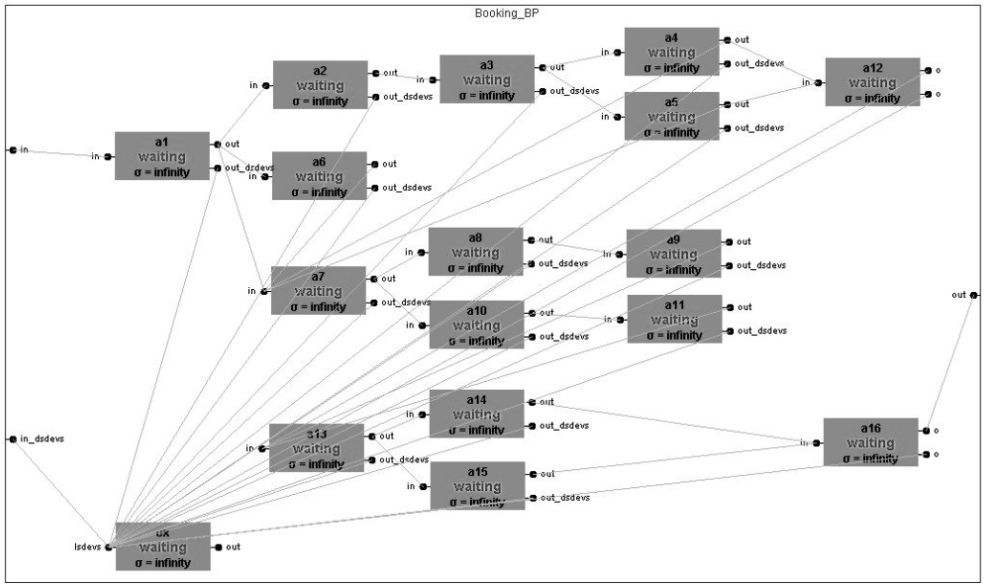


Figure 5. DEVS-Suite Simulation screen shot of the initial DSDEVS model of the Booking BP presented in figure 2.

In order to improve that the generated variants respect the structural constraints in a sound manner, we make redundant simulations with tracking logs (Figure 7). The result log guarantees that the main properties related to the structure constraints are ensured, such as, the deadlock and the lack of synchronization. In figure 7, the results are presented as in a table. Each line represents the simulation of an atomic model related to a given activity inside the business process, where each cell is specific to the state (Phase) transition during the execution time (ms). To make the result clearer we added green circles surrounding the cells where the status is active. The only confirmation of the good execution of the process is the status "terminated" observed

at the last active activity (In this case study, we talk about the activity a16).

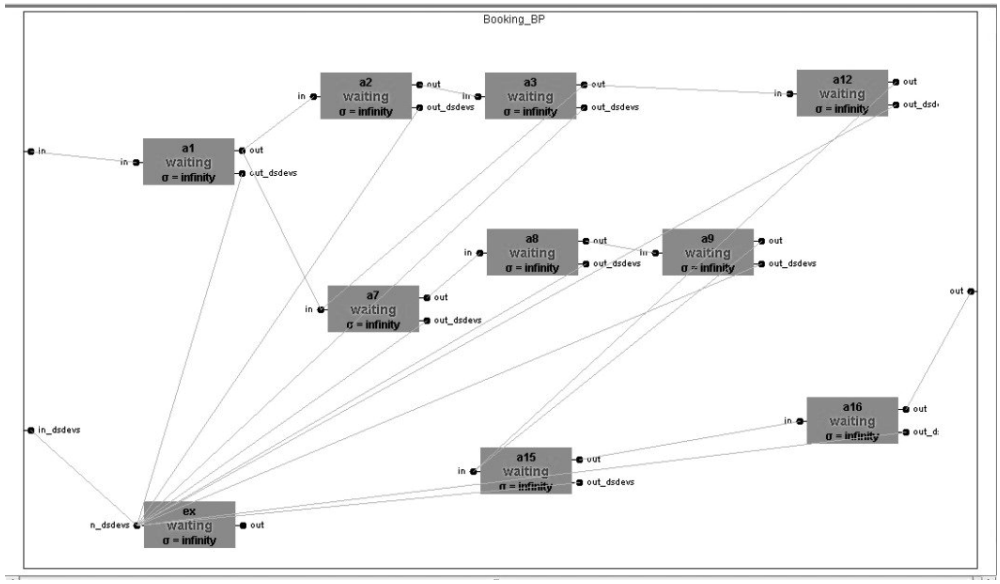


Figure 6. DEVS-Suite Simulation screen shot of the model variant in DSDEVS of the Booking BP presented in figure 2.

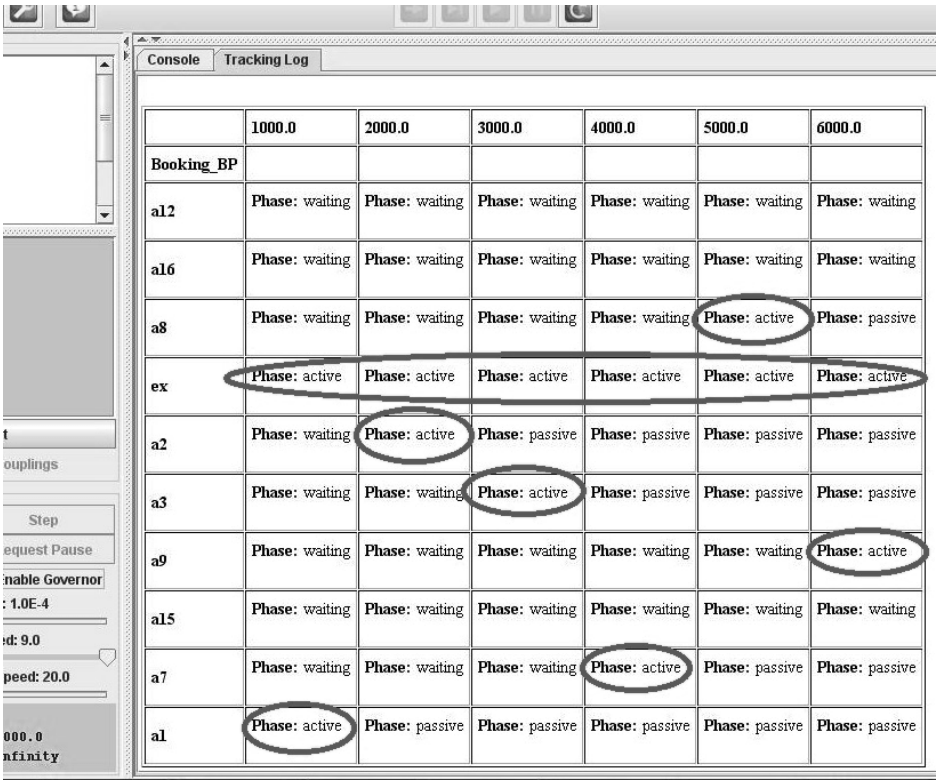
Console | Tracking Log

	1000.0	2000.0	3000.0	4000.0	5000.0	6000.0	7000.0	8000.0	9000.0
Booking_BP									
a1	Phase: active	Phase: passive	Phase: passive	Phase: passive	Phase: passive	Phase: passive	Phase: passive	Phase: passive	Phase: passive
a16	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: active	Phase: terminated
ex	Phase: active	Phase: active	Phase: active	Phase: active	Phase: active	Phase: active	Phase: active	Phase: active	Phase: active
a7	Phase: waiting	Phase: waiting	Phase: waiting	Phase: active	Phase: passive	Phase: passive	Phase: passive	Phase: passive	Phase: passive
a2	Phase: waiting	Phase: active	Phase: passive	Phase: passive	Phase: passive	Phase: passive	Phase: passive	Phase: passive	Phase: passive
a9	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: active	Phase: passive	Phase: passive	Phase: passive
a12	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting
a15	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: active	Phase: passive	Phase: passive
a3	Phase: waiting	Phase: waiting	Phase: active	Phase: passive	Phase: passive	Phase: passive	Phase: passive	Phase: passive	Phase: passive
a8	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: active	Phase: passive	Phase: passive	Phase: passive	Phase: passive

Figure 7. Simulation result of the Booking BP variant in DEVS-Suite with successful context.

We remark that the executive model "ex" is active all the time since it is responsible of all the modifications applied on the BP structure. However, the activities are executed depending to the BP needs. In this example, each activity is executed (phase

= active) at least one time except a12 because of the XOR-split from the outgoing branch of a3: the choice is set to a7 "select flight" instead of a12 "select package". We see that the execution is terminated by the activity a16 with the status "terminated" which explains that set of the temporal constraints is in concordance with the analyst expectations which is reflected with the execution.



	1000.0	2000.0	3000.0	4000.0	5000.0	6000.0
Booking_BP						
a12	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting
a16	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting
a8	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: active	Phase: passive
ex	Phase: active	Phase: active	Phase: active	Phase: active	Phase: active	Phase: active
a2	Phase: waiting	Phase: active	Phase: passive	Phase: passive	Phase: passive	Phase: passive
a3	Phase: waiting	Phase: waiting	Phase: active	Phase: passive	Phase: passive	Phase: passive
a9	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: active
a15	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting	Phase: waiting
a7	Phase: waiting	Phase: waiting	Phase: waiting	Phase: active	Phase: passive	Phase: passive
a1	Phase: active	Phase: passive	Phase: passive	Phase: passive	Phase: passive	Phase: passive

Figure 8. DEVS-Suite simulation result of the Booking BP variant with blocking situation.

If the simulation conducts, unlikely, to a blocking structural situation, such as, a deadlock or a lack of synchronization, regarding to a certain constraints; the proposed model detects the failure and offers to the analyst the way to reconfigure another efficient and reliable variant of the configurable business process. For example, in figure 8, we consider the simulation result of anther variant of the same configurable BP presented in figure 3. We still assume that the outgoing branch of the activity a3 is configured to a XOR-split: this implies that, exclusively, one activity will be executed; a12 or a7. However both of input branches of the activity a15 "cash payment" are configured to AND-join, which means that a15 will be executed if only the both branches are activated (See figure 4). This situation conducts to a deadlock. In the simulation result illustrated in figure 8, the lines representing the activities a15 and a16 show that these models are still waiting for an event to start their execution. That means the model is blocked at a15 which is not able to start up. Technically, the last active activity (a9 in this case) will never transit to the status "terminated".

Besides, it is not programmed to generate this state. This situation conducts to a deadlock.

Therefore, the proposed model consists of an improvement tool of configurable business process with dynamic structure when different contexts are present. Indeed, in DEVS framework, an experimental frame is used to perform validation tests. If the behavior of both the model and its system (business process) counterpart are within acceptable tolerance, the model is then declared valid [21].

6. Conclusion

In this paper, we presented a DSDEVS model in order to handle business processes in different contexts. With this model, the analysts will be able to reuse configurable business processes and make simulations on different process variants regarding to different contexts in order to improve that the generated variants respect the structural constraints in a sound manner by means of redundant simulations with tracking logs. The proposed model guarantees the properties consistency related to the structure constraints, such as, the deadlock and the lack of synchronization. Thus, the variants of a configurable business processes can be verified and validated regarding to structural constraints.

In the future work we plan to enhance the proposed model in order to make it able to propose semantic alternatives in case of lack of concordance with structure constraints or in case of execution failure related to inappropriate structural constraints.

References

[1] M. Rosa, W. Aalst, M. Dumas and F. Milani, "Business Process Variability Modeling," *ACM Computing Surveys*, vol. 50, no. 1, pp. 1-45, 2017. Available: https://doi.org/10.1145/3041957 . [Accessed 2 April 2019].

[2] F. Barros, "Dynamic structure discrete event system specification: A new formalism for dynamic structure modeling and simulation," Winter Simulation Conference, 1995, pp. 781-785.

[3] I. Pihir, N. Vrček, and K. T. Pupek, "Challenges of Processes Simulation with Batch Processing Activities in Contemporary Business Process Management Tools," EBR Conference 2014.

[4] J. Zhang, G. Frey, A. Al-Ahmari, T. Qu, N. Wu and Z. Li, "Analysis and Control of Dynamic Reconfiguration Processes of Manufacturing Systems," *IEEE Access*, vol. 6, pp. 28028-28040, 2017. Available: https://doi.org/10.1109/ACCESS.2017.2757044 . [Accessed 1 October 2020].

[5] Y. Koren, "The Emergence of Reconfigurable Manufacturing Systems (RMSs) ". Benyoucef L. (eds) *Reconfigurable Manufacturing Systems: From Design to Implementation. Springer Series in Advanced*

Manufacturing. Springer, Cham. 2020, Available: https://doi.org/10.1007/978-3-030-28782-5_1 . [Accessed 1 October 2020].

[6] M. Pourbafrani, S. J. van Zelst and W. M. van der Aalst, "Supporting Automatic System Dynamics Model Generation for Simulation in the Context of Process Mining," International Conference on Business Information Systems, Springer, Cham, 2020 pp. 249-263.

[7] S. V. Mierlo, H. Vangheluwe, S. Breslav, R. Goldstein and A. Khan, "Extending Explicitly Modelled Simulation Debugging Environments with Dynamic Structure," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 30, no. 1, 2020, pp. 1-25.

[8] H. Sarjoughian and B. Zeigler, "DEVSJAVA: Basis for a DEVS-based collaborative MS environment," SCS International Conference on Web-Based Modeling and Simulation, San Diego, CA, 1998, pp. 29-36.

[9] W. Benallal, M. Barhamgi, D. Benslimane, Z. Maamar, N. Faci, and A. Bellaaj, "A knowledge-based approach to manage configurable business processes," *Concurrency and Computation: Practice and Experience*, Vol. 32 , no. 15, 2020, pp. e4920. Available: https://doi.org/10.1002/cpe.4920 . [Accessed 4 October 2020].

[10] N. Assy, W. Gaaloul and B. Defude, "Mining configurable process fragments for business process design," International Conference on Design Science Research in Information Systems. Springer, 2014, pp. 209-224. Available: https://doi.org/10.1007/978-3-319-06701-8_14 . [Accessed 4 October 2020].

[11] W. van der Aalst, M. Dumas, F. Gottschalk, A. ter Hofstede, M. La Rosa and J. Mendling, "Preserving correctness during business process model configuration," *Formal Aspects of Computing*, vol. 22, no. 3-4, pp. 459-482, 2010. Available: https://doi.org/10.1007/s00165-009-0112-0 . [Accessed 2 April 2019].

[12] M. A. F. M. S. Jan and R. M. Rosemann, "The quest for organizational flexibility: driving changes in business processes through the identification of relevant context," *Business Process Management Journal,* vol. 22, Iss 4, 2016, Available: http://dx.doi.org/10.1108/BPMJ-01-2015-0007 . [Accessed 2 April 2019].

[13] S. Boubaker, K. Klai, H. Kortas and W. Gaaloul, "A Formal Model for Business Process Configuration Verification Supporting OR-Join Semantics, ". OTM Confederated International Conferences *"On the Move to Meaningful Internet Systems"*, 2018, pp. 623-642. Springer, Cham., vol. 11229. Available: https://doi.org/10.1007/978-3-030-02610-3_35 [Accessed 4 October 2020].

[14] B. Souha, M. Amel, G. Mohamed and G. Walid, "An Event-B Based Approach for Ensuring Correct Configurable Business Processes," 2016

IEEE International Conference on Web Services (ICWS), 2016, pp. 460 – 467.

[15] OMG/BPMN, "Business Process Model and Notation 2.0" [Online]. Available: https://www.omg.org/spec/BPMN/2.0/PDF [Accessed 4 October 2020].

[16] W. Yu, C. Yan, Z. Ding, C. Jiang and M. Zhou, "Modeling and Verification of Online Shopping Business Processes by Considering Malicious Behavior Patterns," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 647-662, 2016. Available: https://doi.org/10.1109/tase.2014.2362819 . [Accessed 2 April 2019].

[17] W. Yu, C. Yan, Z. Ding, C. Jiang and M. Zhou, "Modeling and Validating E-Commerce Business Process Based on Petri Nets," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 3, pp. 327-341, 2014. Available: https://doi.org/10.1109/tsmc.2013.2248358 . [Accessed 2 April 2019].

[18] H. Alena, B. Thomas and R. Manfred, "Guaranteeing Soundness of Configurable Process Variants in Provop," 2009 IEEE Conference on Commerce and Enterprise Computing, 2009, pp. 98-105.

[19] Provop, "Managing and Configuring Process Variants" [Online]. Available: https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.030/research/provop/ [Accessed 4 October 2020].

[20] M. Reichert, S. Rechtenbach, A. Hallerbach and T. Bauer, "Extending a business process modeling tool with process configuration facilities: The Provop demonstrator," 2009.

[21] B. Zeigler, H. Praehofer and T. Kim, *Theory of modeling and simulation*. San Diego: Academic Press, 2000.

[22] M. Sbayou, Y. Bouanan, G. Zacharewicz, J. Ribault and J.François, "DEVS modelling and simulation for healthcare process application for hospital emergency department," *Proceedings of the 50th Annual Simulation Symposium*, 2017, pp. 1-12.

[23] Y. Bouanan, G. Zacharewicz, B. Vallespir, J. Ribault and S. Y. Diallo, (2016). "DEVS based network: modeling and simulation of propagation processes in a multi-layers network," *Proceedings of the Modeling and Simulation of Complexity in Intelligent, Adaptive and Autonomous Systems 2016 (MSCIAAS 2016) and Space Simulation for Planetary Space Exploration (SPACE 2016)*, pp. 1-8.

[24] W. A. E. Manzano, V. V. G. Neto and E. Y. Nakagawa, "Simulating Systems-of-Systems Dynamic Architectures," *Revista Eletrônica de Iniciação Científica em Computação*, *17*(2), 2019.

[25] S. Boukelkoul and R. Maamri, "Optimal model transformation of BPMN to DEVS," 12th ACS/IEEE International Conference on Computer Systems and Applications AICCSA, IEEE Computer Society, 2015, pp. 1-8.

[26] DEVS-Suite [Online]. Available: https://acims.asu.edu/software/devs-suite/ [Accessed 4 October 2020].

[27] H. Zhang, W. Han and C. Ouyang, "Extending BPMN for Configurable Process Modeling," *Moving Integrated Product Development to Service Clouds in the Global Economy*, ISPE, 2014, pp. 317–330.

[28] M. Rosemann and W. van der Aalst, "A configurable reference modelling language," *Information Systems*, vol. 32, no. 1, pp. 1-23, 2007. Available: https://doi.org/10.1016/j.is.2005.05.003 . [Accessed 2 April 2019].

[29] A. Nour and G. Walid, "Extracting Configuration Guidance Models from Business Process Repositories," International Conference on Business Process Management, 2016, pp. 192-206.