A PROPOSAL OF USING DEVS MODEL FOR PROCESS MINING

Yan Wang, Grégory Zacharewicz, David Chen, Mamadou Kaba Traoré

A PROPOSAL OF USING DEVS MODEL FOR PROCESS MINING

Yan Wang[(a)], Grégory Zacharewicz[(b)], David Chen[(c)], Mamadou Kaba Traoré[(d)]

[(a),(b),(c)] IMS, University of Bordeaux, 33405 Talence Cedex, France
[(d)] LIMOS, Université Blaise Pascal, 63173 Aubiere Cedex, France

[(a)]yan.wang@etu.u-bordeaux.fr, [(b)]gregory.zacharewicz@u-bordeaux.fr,
[(c)]david.chen@ims-bordeaux.fr, [(d)]traore@isima.fr

ABSTRACT
Process mining is a relative young research area which consists of process modeling and data mining. Process discovery as a part of the process mining focuses on converting event logs into process models. Petri Nets formalism is identified as the most convenient resulting process model. However, it is not entirely satisfying and needs to be improved with the purpose of covering the temporal aspects of the studied system. Compared with it, DEVS has the advantage of explicit and concurrent time and separating model from simulation. The objective of this paper is to specify DEVS model as the resulting process model. Based on the existing Two-Phased Approaches in process mining, a region-based approach with the suitable mapping is designed to convert the transition system directly to DEVS. A study case is presented to implement this approach. This paper is a position paper and it needs to be completed. In addition, a dynamic semantic should be designed to solve some typical representational limitations and a simulation engine should be selected.

Keywords: process mining, process discovery, DEVS, dynamic semantic, time

1. INTRODUCTION

Nowadays, the development of the information systems cannot be separated by the operational process. Moreover, more and more events with abundant process information are recorded. Despite the omnipresence of event data, most of the organizations analyze and build models based on expert assumption rather than quantified collected fact. Thanks to process mining [Van der Aalst, 2011], it provides the methods to discover monitor and improve actual processes by using event data to extract process-related information. Process mining combines process design by model-driven approaches and data mining. It includes three types: process discovery, process conformance and process enhancement. This paper anticipates and lays a cornerstone to make a breakthrough on the process discovery area.

The purpose of discovery technique is to extract an event log with using recorded historical information and produce a model for simulation. In the process mining, many models can be selected as the resulting model for business process for example workflow nets and BPMN, whereas Petri Nets [Peterson, 1977] is frequently identified as the direct resulting process models because Petri nets is formal, simple and graphical while still allowing for the modeling of concurrency, choices, and iteration. When considering about the representational bias such as concurrency, loops and even OR-splits/joins, not every process model can be totally satisfied. Petri Nets is able to discover the concurrency in spite of OR-splits. As there are many potentially concurrent activities, Petri Nets may either reach some limits in describing time or distinguish information in the process. The Discrete Event System Specification (DEVS) [Zeigler et al., 2000] provides a hierarchical and modular formalism to describe a state and event based system. It contains the basic components like explicit time, inputs, outputs, states and functions. The characteristic of separating model and simulation makes DEVS able to demonstrate the same function in Petri Nets. The semantic execution of DEVS can satisfy various potential requirements, such as distinguishing different information processing, and solve representational limitations. So we are trying to design a new approach to discover DEVS model as the resulting process model for business process simulation.

The paper is organized as follow. Section 2 provides the background of DEVS formalism in comparison with the Petri net and an introduction about transition systems. Section 3 gives the introduction about process discovery, XES standard and DEVS related research. Two-phased approach is also presented. Section 4 proposes to construct a complex and meaningful modeling and simulation structure and presents the methodology design with the mapping. Section 5 makes the simulation based on a study case. Section 6 shows the limitation and indicates the future works. The paper ends with a conclusion.

2. BACKGROUND

2.1. DEVS Formalism

The DEVS formalism for modeling and simulation is based on discrete events, and provides a framework with mathematical concepts based on the set theory and

the system theory concepts to describe the structure and the behavior of the system [Zeigler et al., 2000]. A real system modeled by DEVS is described as a number of connected behavioral (atomic) and structural (coupled) components. A DEVS atomic model is formally defined as:

$$M = <X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta>.$$

Where X is the set of input values; S is the set of states; Y is the set of output values; δ_{int} is the internal transition function; δ_{ext} is the external transition function; λ is the output function; ta is the duration function. From the semantics of DEVS formalism [Zeigler et al., 2000], we know that each atomic model has the duration specified by ta(s). When the elapsed time e = ta(s), the state duration expires and the atomic model will send the output $\lambda(s)$ and performs an internal transition to a new state specified by $\delta_{int}(s)$. However, state transition can also happen due to arrival of an external event which will place the model into a new state specified by $\delta_{ext}(s,e,x)$; where s is the current state, e is the elapsed time, and x is the input value. The time advance function ta(s) can take any real value from 0 to ∞. A state with ta(s) value of zero is called transient state, and on the other hand, if ta(s) is equal to ∞ the state is said to be passive, in which the system will remain in this state until receiving an external event.
The graphical notation is also used for simulation in this paper [Song et al., 1994]. Each node represents an activity or a state, dotted arc denotes an internal transition and solid arc denotes an external transition. The output events in both internal transition and external transition have different labels. An output event p!m means that a message m is output at the port p. Similarly, an input event p?m means that a message m is input at the input port p.
A coupled DEVS model consists of atomic and other coupled models connected together. They are formally defined as:

$$N = <X, Y, D, EIC, EOC, IC>.$$

Both X and Y respectively define the sets of input and output events. D is a set of indices for the components. The external input coupling (EIC) specifies the connections between external and component inputs, while the external output coupling (EOC) describes the connections between component and external outputs. The connections between the components themselves are defined by the internal coupling (IC). The coupling and transformation between separating atomic models make it possible to construct a more complicated hierarchical model.

2.2. Petri Nets

Petri Nets is a modeling formalism originally developed by C.A Petri [Peterson, 1977]. A Petri net is a bipartite graph consisting of place and transition. Arc is used to connect between place and transition. The network structure is static and token flows through the network. Despite the four parameters, the state of a Petri net is determined by the distribution of tokens and is referred as marking. Enabling and firing are the main operating rules. Comparing with DEVS, Petri Nets can be embedded into DEVS because a DEVS model can represent any discrete event behavior. Also, DEVS model has the timing characteristics connected with the reality whereas Petri Nets cannot present. Zeigler [Zacharewicz et al., 2008] discusses how DEVS modeling is more accurate than Petri nets modeling due to the following facts.

- DEVS gives a more general framework for modeling and simulation of complex systems.
- DEVS integrates naturally the notion of time contrary to Petri nets which require an extension of the formalism.
- DEVS offers a formal (and separated from model) definition of the simulator.

2.3. Transition Systems

Transition systems are considered as the original transformation process model in this paper. Robert [Robert, 1976] gives a formal definition about a transition system. It is a pair $(S, \rightarrow)$ where S is a set of states and $\rightarrow$ is a binary relation on S, called the set of transitions. A named transition system is a triple $(S, \rightarrow, \Sigma)$ where $(S, \rightarrow)$ is a transition system and each transition system is assigned one or more names in the set Σ. A visualized transition system [Van der Aalst, 2011] consists of states and transitions. The states are represented by black circles. There are one initial state and one final state. Each state has a unique label. This label is merely an identifier and has no meaning. Transitions are represented by arcs. Each transition connects two states and is labeled with the name of an activity. Multiple arcs can bear the same label.
Given a transition system one can reason about its behavior. The transition starts in one of the initial states. Any path in the graph starting in such a state corresponds to a possible execution sequence. A path terminates successfully if it ends in one of the final states. A path deadlocks if it reaches a non-final state without any outgoing transition. The transition system may live-lock if some transitions are still enabled but it is impossible to reach one of the final states.

3. THE STATE OF THE ART

3.1. Process discovery

Process discovery as a part of process mining is the main area we want to develop. General process discovery problem [Van der Aalst, 2011] is defined like this. Let L be an event log specified by the XES standard. A process discovery algorithm is a function that maps L onto a process model such that the model is "representative" for the behavior seen in the event log. The challenge is to find such an algorithm.
Until recently, the de facto standard for storing and exchanging events logs was MXML (Mining eXtensible

Markup Language). MXML emerged in 2003 and was later adopted by the process mining tool ProM. XES is the successor of MXML. Based on many practical experiences with MXML, the XES format has been made less restrictive and truly extendible. An XES document (i.e., XML file) contains one log consisting of any number of traces. Each trace describes a sequential list of events corresponding to a particular case. The log, its traces, and its events may have any number of attributes. To provide semantics for such attributes, the log refers to so-called extensions.

There are many different approaches to do the actual discovery. Two-Phase Approaches is one of the identified approaches. It first constructs a "low-level model" and then converts into a "high-level model" and other more advanced control-flow patterns. There are four steps in this approach: extract event logs; create a transition system based on one abstraction; convert the transition system into a Petri net; convert the Petri net into other notations (e.g., BPMN). In this new approach, we reuse the first two steps as the input and design a new approach of the transformation between transition system and DEVS model.

3.2. DEVS related research

DEVS can be easily transformed from Petri Nets or BPMN although we don't need this transformation. Jacques and Wainer [Jacques et al., 2002] propose an approach of mapping of the Petri Net modeling formalism into the DEVS modeling formalism using an unmodified DEVS simulator. They also show that DEVS simulation results can simply be filtered through a parsing tool to give them a stronger PN flavor. Bazoun et al. [Bazoun et al., 2014] define a transformation approach of BPMN models into DEVS simulation models based on the meta-model approach and describe the enrichment of obtained DEVS models through performance indicators. This approach includes an exhaustive mapping, the transformation architecture and an implementation in SLMToolBox M&S tool. Zacharewicz et al. [Zacharewicz et al., 2008] describe a language for workflow processes and a new transformation algorithm from Workflow XML specification to G-DEVS model. This language supports algorithms to transform the Workflow model.

Some other researches also focus on discovering the interoperability of DEVS. Wainer et al. [Wainer, 2009] standardize the simulation middleware to interface different simulation environments and allow synchronization for the same simulation run across a distributed network regardless of their model representation. They provide several approaches for example DEVS/SOA distributed simulation platform, DDSP, the shared abstract model, RISE and DEVS namespaces which standardize DEVS simulation middleware in different ways.

DEVS has already been widely used in many areas. Until now, there are many professional descriptions about its application. Song et al. [Song et al., 1994] introduces concepts of inverse DEVS and defines controllability of discrete event systems expressed in the DEVS formalism. A graphical notation is presented to visualize DEVS models. The concurrency is also discussed to analyze the dynamics of a system consisting of several subsystems. Zeigler and Sarjoughian [Zeigler et al., 2005] present the systems entity structure/model base (SES/MB) framework for simulation process. First it sets up a DEVS model base as the organized libraries. There are FIFO, generator, transducer and processor model inside the model base. The knowledge of the desired system is represented by the SES. Then through retrieving component and coupling them together, we can get the synthesized model. At last, this model is evaluated via simulation. This framework can be identified as the basis to construct the semantic of the proposition.

4. DESIGN AND IMPLEMENTATION

4.1. Approach design

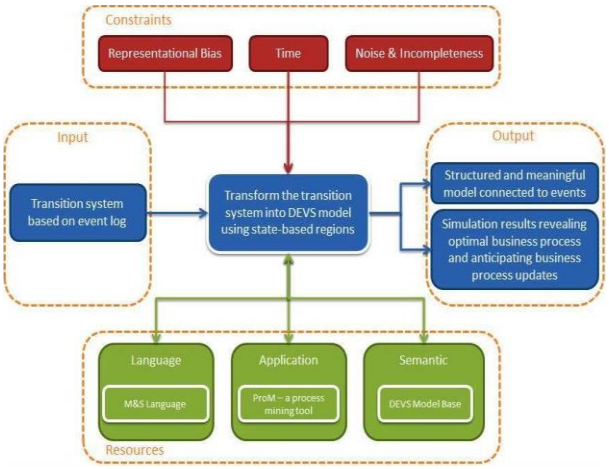

Figure 1: Proposed framework

Figure 1 shows the general view of the proposed framework. We reuse the discovery technique and take the transition system instead of Petri Nets or BPMN as input. In the resource part, we need to provide the DEVS model base to support the new approach. As the reuse of the discovery technique is operated in the process mining tool ProM, we continue to use it in the new approach. A lot of plug-ins implementing various techniques can be used in the ProM and it is a challenge to design a new plug-in for the DEVS model. The representational bias, noise and incompleteness appear when we try to transform the event logs into other formalism because the result isn't always optimal and desired. Furthermore, it will limit the search space of simulation. According to the representational bias, parallel DEVS models with multiple inputs and outputs and time segments can solve the problem of concurrency, a switch network can solve the problem of OR-splits, generator coupled model can solve the problem of loop and duplicate actions. All these models will be designed in the DEVS model base. As the process discovery is tightly connected with the reality,

more and more limitations will describe the enrichment of DEVS model base semantic. The noise defines event logs contain rare and infrequent behavior not representative for the typical behavior of the process. The incompleteness is defined as that the event logs contain too few events to be able to discover some of the underlying control-flow structures. Besides, time is another very important constraint which will lead to desired state scheduling. A time-related approach will be designed to make selections on transitions according to the semantics of DEVS formalism. At last, we wish to implement a structured and meaningful model used for anticipating optimal business process.

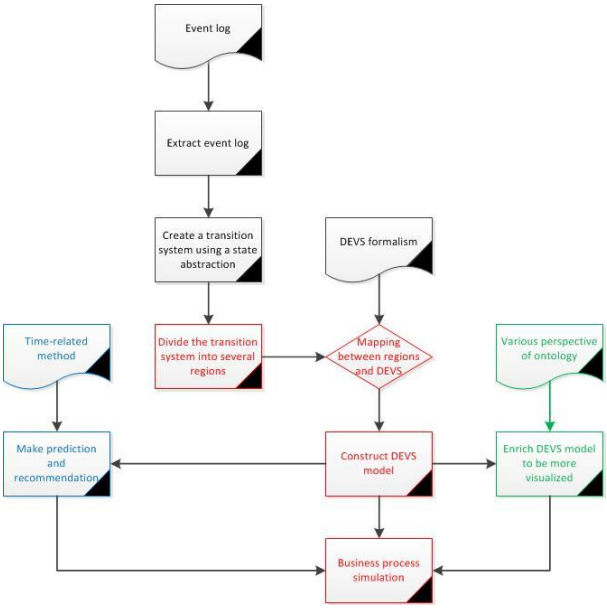

Figure 2: Process of mapping transition system to DEVS model

Figure 2 shows the main process of Modeling & Simulation by integrating DEVS model with process mining. The black part has already been done in the process mining and the red part is the main work of this paper. First we need to extract the useful information from the list of event logs. Using the state abstraction, we can automatically construct a transition system based on the event logs. The main part of this approach is to implement the transformation between transition system and DEVS model. To start to implement the region-based approach, we divide the transition system in several regions. In this case all activities can be classified into entering the region, leaving the region and non-crossing. An activity cannot be entering the region in one part of the transition system and exiting the region in some other part. The mapping in the next section makes contribution on implementing this transformation and a new DEVS model is discovered. The discovered DEVS model will be used for business process simulation. The green part is to extend the ontology of the DEVS model. It is necessary to construct the DEVS model base. As DEVS model has the characteristic of separating model simulation, each region can be mapped with one component in the model base. It has atomic model and coupled model with special functions which we have talked in the last paragraph. These models will be used as components to support the transformation. Moreover, we need to extend various perspectives (organizational perspective, time perspective, data perspective, etc.) based on ontology in order to be more meaningful and visualized like a map. The blue part is to design a time-related approach in order to predict the desired process flow and provide a more intuitive visualization. As the time is not considered in the transition systems, it is necessary to design a method to obtain the time directly from event logs.

4.2. Mapping design

From Two-Phased Approaches, we know that it converts the transition systems to the Petri Nets. Table 1 shows the mapping between transition systems and Petri Nets. Transition systems are divided by several regions and each region is converted into Place in Petri Nets. Transitions in the transition systems are used to create transition in Petri Nets with the name of activity. Also, transition and place are connected by arc which constructs the whole structure of Petri Nets. If the activities are entering into the region, we set the arc from transition to place. If the activities are leaving the region, we set the arrow from place to transition. Only the first place has the marking which is converted from the initial state.

Table 1: Mapping between transition systems and Petri Nets

Transition Systems	Petri Nets
Region	Place
Transition	Transition
Activity	
State	Marking

Based on table 1, we design the mapping between transition systems and DEVS as illustrated in table 2. Region is transformed to atomic model in DEVS. Each atomic model has a unique label as identify. The initial state is also transformed into the first atomic model. We extract the activities from transition system and transform into the associated input and output event. As the transition in transition systems contains the name and the value of transition, we take the name as the port of each atomic model and the value of transition as the message. To avoid the name repeat, we combine the name of source and target as the name of the transition. If the message is related to “start”, we identify it as internal transition with output event. If the message is related to “finish”, we identify it as external transition with input event. The transition is visualized by arc and connects the source and target atomic model. In the future work, as the transition system don’t have the information about time, we extract the time directly from event logs and set time duration in atomic model by using time related approach. Each execution of the transition is decided by the time and such visualizations

are important to get insight into the desired process flow.

Table 2: Mapping between transition systems and DEVS

Transition system	DEVS
Region	Atomic model
Transition	Internal transition
	External transition
Activity	Input event
	Output event
State	State

4.3. Time concurrency

The transition system in figure 3 was obtained from log [{a, b, c, d}^{3}, {a, c, b, d}^{2}, {a, e, d}]. This demonstrates that region-based approach can be used to convert transition systems to DEVS by using mapping. Consider for example Region R1 = {[a], [a, b]}. All a labeled transitions in the transition system enter R1 (there is only one), all c labeled transition exit R1 (there are two such transitions), all e labeled transition exit R1 (there are only one), and all other transitions in the transition system do not cross R1. Hence, R1 is a region corresponding to atomic model M2 with input event and output event c and e. In the transition systems, there are six minimal regions and b and c are concurrent. However, in the DEVS model every atomic model is discrete and independent model. Every atomic model has several time durations with associated transition. It shows a strong concurrency because each transition is controlled by the time. For example b and c is OR-split relationship in transition systems and it means we don't know which one happen first. But in the DEVS model, M1 has time duration t1 and M2 has time duration t2. If t1 is bigger than t2, c24?complete will execute first. If t1 is smaller than t2, b13?complete will execute first. This demonstrates that each transition in DEVS is independent corresponding to the time duration with a unique label. The discrete modeling simulation can solve many complicated process discovery.

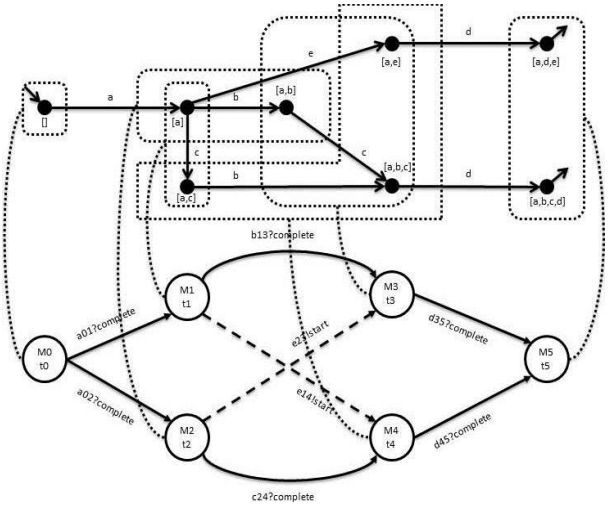

Figure 3: Transition system is converted into DEVS using mapping

5. STUDY CASE

A list of event logs is used as input for process discovery as shown in figure 4. An event can have any number of attributes. However, an extension gives semantics to particular attributes. There are four extensions in these event logs. The concept extension defines the name attribute for traces and events. The lifecycle extension defines the transition attribute for events. Time extension defines a timestamp attribute and both the data and time are recorded. The organizational extension defines a resource attribute. The resource attribute refers to the resource that triggered or executed the event. This example log in figure 4 specifies two lists of global attributes (one is hidden). Traces have one global attribute: attribute concept:name is mandatory for all traces. Events have two global attributes: attributes lifecycle:transition and concept:name are mandatory for all events. It also defines three classifiers in these event logs. Each classifier is specified by a list of attributes. There are four traces in these event logs but only one is visible. Case 1 has four events and each event has four attributes. When mining transition systems starts, the event name and transition is classified as activities and transformed into transition. Time is not considered and resource is all undefined. The information is classified and extracted by the plug-in openXES based on XES standard.

```
<?xml version="1.0" encoding="UTF-8"?>
- <log xmlns="http://www.xes-standard.org/" openxes.version="1.0RC7" xes.features="nested-attributes" xes.version="1.0">
    <extension uri="http://www.xes-standard.org/lifecycle.xesext" prefix="lifecycle" name="Lifecycle"/>
    <extension uri="http://www.xes-standard.org/org.xesext" prefix="org" name="Organizational"/>
    <extension uri="http://www.xes-standard.org/time.xesext" prefix="time" name="Time"/>
    <extension uri="http://www.xes-standard.org/concept.xesext" prefix="concept" name="Concept"/>
    <extension uri="http://www.xes-standard.org/semantic.xesext" prefix="semantic" name="Semantic"/>
  - <global scope="trace">
      <string value="__INVALID__" key="concept:name"/>
    </global>
  + <global scope="event">
    <classifier name="MXML Legacy Classifier" keys="concept:name lifecycle:transition"/>
    <classifier name="Event Name" keys="concept:name"/>
    <classifier name="Resource" keys="org:resource"/>
    <string value="Rapid Synthesizer" key="source"/>
    <string value="excercise3.mxml" key="concept:name"/>
    <string value="standard" key="lifecycle:model"/>
  + <trace>
  + <trace>
  + <trace>
  - <trace>
      <string value="Case1.0" key="concept:name"/>
    - <event>
        <string value="UNDEFINED" key="org:resource"/>
        <date value="2008-12-09T08:20:01.527+01:00" key="time:timestamp"/>
        <string value="A" key="concept:name"/>
        <string value="complete" key="lifecycle:transition"/>
      </event>
    - <event>
        <string value="UNDEFINED" key="org:resource"/>
        <date value="2008-12-09T08:21:01.527+01:00" key="time:timestamp"/>
        <string value="C" key="concept:name"/>
        <string value="complete" key="lifecycle:transition"/>
      </event>
    - <event>
        <string value="UNDEFINED" key="org:resource"/>
        <date value="2008-12-09T08:22:01.527+01:00" key="time:timestamp"/>
        <string value="E" key="concept:name"/>
        <string value="complete" key="lifecycle:transition"/>
      </event>
    - <event>
        <string value="UNDEFINED" key="org:resource"/>
        <date value="2008-12-09T08:23:01.527+01:00" key="time:timestamp"/>
        <string value="G" key="concept:name"/>
        <string value="complete" key="lifecycle:transition"/>
      </event>
    </trace>
</log>
```

Figure 4: XES document for event logs

After these event logs are extracted, we get the mined transition system as shown in figure 5. It has an initial state represented by dotted line and an acceptance state represented by double line. Each state has an identity label and each transition has the label of activity. The state and the transition are connected to provide several possible execution sequences.

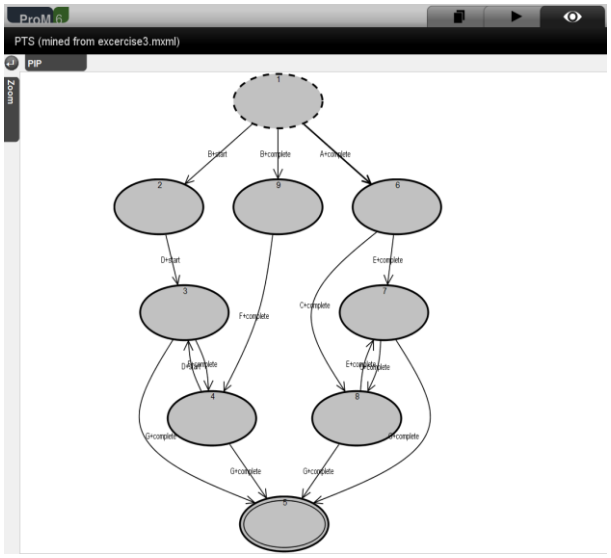

Figure 5: Mined transition system in ProM

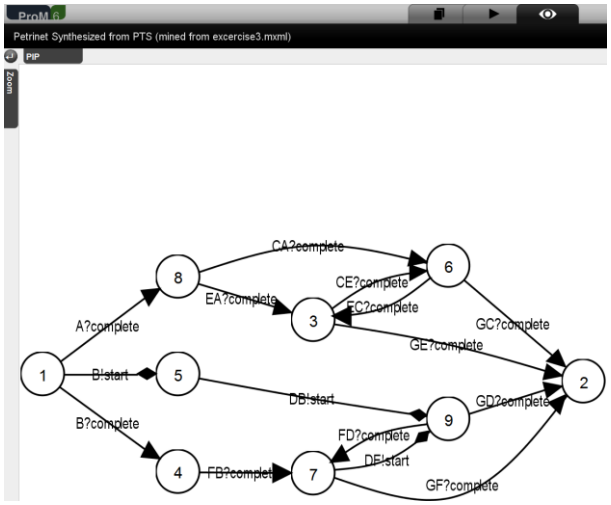

Figure 6: DEVS model is converted from transition system using region-based approach in ProM

After we get the transition system, we start to use the region-based approach to transform transition system into DEVS as shown in figure 6. We transform the components in the transition system to the related DEVS components according to the mapping. First we check if the transition system meets the forward-closure property. Transition system is divided into regions and then transformed to atomic model. Each atomic model has a unique identity number and each transition has a unique name which represents event and message. External transition which contains the message "complete" is visualized as a classical arrow and internal transition which contains the message "start" is visualized as a diamond arrow. The name of the transition is the combination of the source activity and the target activity. Each state (we call here state, the state variable phase) with a leaving internal transition is given a time life function not infinite to be defined. Each state with no internal transition has its time life function set to infinite. The initial state is converted into the atomic model with label of 1. In this study case the time in the atomic model is not extracted from the event logs. The business process starts with the first atomic model and continues the transitions automatically until it reaches the desired state.

6. PERSPECTIVE

Region-based approach has reached the first step to integrate DEVS model in the process mining. However, it is still a big challenge to design DEVS for business process simulation. How to satisfy the customer's requirement for simulation is the main objective. This section gives the limitation of the recent work and makes the perspective for the future work.

The DEVS model in the study case is not executed. As the time is not considered, the transition from one state to another state is not controllable. It is necessary to design a method to extract time from event log and integrate with atomic models. If every atomic model has the time duration, the execution of each transition will depend on the semantics of DEVS formalism. The time perspective is concerned with the timing and frequency of events. The presence of timestamps enables the discovery of bottlenecks, the monitoring of resource utilization, and the prediction of remaining processing times of running cases.

Until now, DEVS model in the process mining cannot be used directly for simulation. The graphical view of process model needs to be enriched with temporal information at least very basic one that are time life function values and event planning. After having enriched the model, there are two methods to run it. The first one is to implement the interoperability with other DEVS simulation engine platform like ADEVS, CD++ and so on. We need to export the DEVS model from the ProM and transform into other pattern which another platform can accept. So the discovered DEVS can be simulated in an existing platform. The second one is to implement a DEVS simulation engine inside ProM. Thanks to conformance checking, it defines that the behavior of a process model and the behavior recorded in an event log are compared to find commonalities and discrepancies. So DEVS not only can be used for business simulation, but also it can reduce the deviations from the reality. Deviations from reality include model deviating from reality and case deviating from model. Conformance checking supports deviation by providing algorithms for example token replay.

Moreover, in the DEVS model, the atomic model is not explicitly presented. Different atomic models with different functions need to be visualized. A dynamic and abundant DEVS model base semantic will be constructed with the purpose of reducing the representational bias. The DEVS model base is used as a component library in order to construct complicated DEVS models. The enrichment of DEVS model will also use ontology to extend various perspectives. Discovered models may focus on different perspectives (control-flow, data flow, time, resources, costs, etc.) and show these at different levels of granularity and precision.

7. CONCLUSION

In this paper, we presented the current process discovery technique which extracts event logs to discover models. The novelty is coming from DEVS and it is specified as the resulting model for business process simulation. In comparison with Petri Nets, DEVS uses explicit time and separates model from simulation. We integrated DEVS model with process discovery to build an advanced business process model. To achieve this goal, we selected the transition systems as input after extracting event logs. A region-based DEVS transformation approach with a suitable mapping has been designed. Meanwhile, we highlight the advantage of time concurrency in DEVS model. An example is presented to show this approach by using ProM simulation platform. According to the perspective, DEVS models generated need to be improved with the time information. Then a simulation engine has to be selected and finally, we assume that the use of ontology areas will help to improve the model description. So users will be able to build almost automatically the business process model, make simulations, identify various components and supervise the operating process.

REFERENCES

Bazoun H., Bouanan Y., Zacharewicz G., Ducq Y., Boye H., 2014. Business Process Simulation: Transformation of BPMN 2.0 to DEVS Models. Proceedings of the Symposium on Thery of Modeling & Simulation, Spring Simulation Multi-conference, No. 20.

Bouanan Y., El Alaoui, M.B., Zacharewicz, G., and Vallespir B., 2014. Using DEVS and Cell-DEVS for modelling of information impact on individuals in social network. In Advances in Production Management Systems. Innovative and Knowledge-Based Production Management in a Global-Local World, 409-416. Springer.

Jacques C.J.D., Wainer G.A., 2002. Using the CD++ DEVS Tookit to Develop Petri Nets. Summer computer simulation.

Peterson J.L., 1977. Petri Nets. ACM computing surveys, 9(3), 223-252.

Robert M.K., 1976. Formal verification of parallel programs. Communications of the ACM, 19(7), 371-384.

Song H.S., Kim T.G., 1994. The DEVS framework for discrete event systems control. Proceedings of the Fifth Annual Conference on Distributed Interactive Simulation Environment, 228-234. IEEE.

Van der Aalst W.M.P., 2011. Process mining: discovery, conformance and enhancement of business processes. Springer Science & Business Media.

Wainer G.A., 2009. Discrete-event modeling and simulation: a practitioner's approach. New York: CRC Press.

Zacharewicz G., Frydman C., Giambiasi N., 2008. G-DEVS/HLA Environment for Distributed Simulations of Workflows. Society for computer simulation international, San Diego, CA, USA, 84 (5), 197-213.

Zeigler B.P., Praehofer H., and Kim T.G., 2000. N-dimensional Cell-DEVS models. Discrete Event Dynamic Systems, 12(2), 135-157.

Zeigler B.P., Praehofer H., and Kim T.G., 2000. Theory of Modeling and Simulation. Academic Press: New York, USA.

Zeigler B.P., Sarjoughian H.S., 2005. Introduction to DEVS Modeling and Simulation with JAVA: Developing Component-Based Simulation Models. Technical Document, University of Arizona, 129-147.