Modeling and Simulation Versions of Business Process using Petri Nets

Fatma Ellouze, Mohamed Amine Chaâbane, Rafik Bouaziz
University of Sfax/ Faculty of Economics and Management of Sfax
Route de l'aéroport, BP1088 Sfax, Tunisia
fatma_ellouze@hotmail.com; {Ma.chaabane, Raf.bouaziz}@fsegs.rnu.tn

Eric Andonoff
University of Toulouse/ IRIT
2 Rue de Doyen Gabriel Marty, 31042 Toulouse Cedex, France
andonoff@univ-tlse1.fr

Abstract — This paper proposes a solution for modeling and simulating flexible Business Processes (BP) using version concepts. It advocates a Model Driven Architecture approach to handle versions of BP. The paper presents, first, a meta model (VBP2M: Versioned Business Process Meta model) for modeling versions of BP considering six perspectives of business processes: process, functional, operation, informational, organizational and intentional perspectives. Then, it proposes an extension of the meta model of Petri nets (PN) to support the version concepts. Finally, it defines mapping rules to translate versions of BP modeled in accordance with the VBP2M to PN models, with graphical representations, in order to be able to simulate the behavior of each version of BP.

Keywords-Flexible Business Process; Version; VBP2M; Petri nets; Mapping rules; MDA framework.

I. INTRODUCTION

Nowadays, the importance of Business Processes (BP) in organizations' Information Systems (IS) is widely recognized and these last few years, there has been a shift from data-aware IS to process-aware IS [1] [2] [3]. However, despite important advances in Business Process Management (BPM), several issues are still to be addressed. Among them, the business process flexibility issue is a really a relevant and challenging one. Indeed, the competitive and dynamic environment in which organizations and enterprises evolve leads them to frequently change their business processes in order to meet new production or customer requirements, new legislation or business rules. We define flexibility of BP as *the ability of BP to deal with both foreseen and unforeseen changes in the environment in which they operate* [4].

This issue has mainly been addressed using two main approaches: a declarative (decision oriented) approach and a procedural (activity oriented) approach. The declarative approach consists in defining a set of constraints defining BP execution rules [5] [6] [7] [8]. In this approach, dealing with flexible BP seems very easy as we just need to add and/or remove constraints to define new BP execution rules. However this approach is only available in the Declare BPM suite [13], which is an academic solution unknown in the industry. At the opposite, the procedural approach is widely used in industrial Workflow management systems and BPM suites. Thus BPM community has to deal with procedural BP flexibility. In this approach, a BP is defined as a set of activities coordinated by using control patterns [9]. Several techniques have been introduced to address BP flexibility and among them we quote late binding, late modeling and versioning [10] [11].

In a previous work [12], we advocated to use versioning for BP flexibility. Indeed, using this technique, it is possible to deal with the different flexibility types defined in [11] as it is possible to handle, at the same time, different schemas (versions) of a given BP.

Our proposition [12] extends the three main contributions about BP versions ([16] [17] [18]) considering, in addition to the process and functional perspectives of BP. four other perspectives aiming to have a comprehensive description of BP [9] [10]. These perspectives are: (i) the operation perspective which defines actions to be achieved within an atomic activity, (ii) the informational perspective which describes the structure of information consumed and/or produced by the BP, (iii) the organizational perspective which details roles, organizational units and actors invoked by the BP and (iv) the intentional perspective which explains the context of use of a BP.

To sum up, our previous works [19][20] deal with BP flexibility issue adopting the procedural paradigm and using the versioning technique. It introduced VBP2M (Versioned Business process Meta model) for BP version modeling. However, we did not addressed the simulation and verification of the modeled BP versions in order to check their behavior. As a consequence, the aim of this paper is to simulate and verify the behavioral dimension of the modeled versions of BP using conventional Petri nets. Conventional Petri nets have been chosen since they are recognized as a perfect mean for simulating and verifying distributed applications and they are widely used in BPM [14] [15]. More precisely, this paper:

- advocates an MDA approach for dealing with BP simulation;
- extends PN meta model to integrate BP version concepts;
- defines a transformation process including (i) the mapping between concepts of VBP2M and extended PN meta model and (ii) translation related rules;
- presents a tool implementing this transformation.

The remainder of this paper is organized as follows. Section 2 shows how we model versions of BP. Section 3 introduces the MDA framework we propose to handle versions of BP. Firstly, this section details the three models (CIM, PIM and PSM) of the framework. Secondly, it gives mapping rules allowing to represent a version of BP, modeled according the previous meta model, as a tree. Thirdly, it explains mapping rules to translate a version of BP represented as a tree to a Petri Net model. Section 4 illustrates our proposals within a case study. Section 5 presents the tool implementing our contributions. Finally, Section 6 recaps our contributions, discusses them according related work, and gives some perspectives for our future works.

II. MODELING VERSIONS OF BUSINESS PROCESS

This section briefly presents the Versioning Business Process Meta model (VBP2M) we have proposed to model versions of business process [12][19][20]. More precisely, first it introduces the version concept and then it details the VBP2M for versions of BPs.

A. The version concept

A real world entity characteristic that may evolve during its life cycle: it has different successive states. A version corresponds to one of the significant entity states. So, it is possible. Hence, it is possible to manage several entity states. The entity versions are linked by derivation link; they form a derivation hierarchy. When created, an entity is described by only one version. The definition of every new entity version is done by derivation from a previous one. Such versions are called derived versions. Several versions may be derived from the same previous one. They are called alternative version. Fig. 1 illustrates a derivation hierarchy to describe entity evolution.

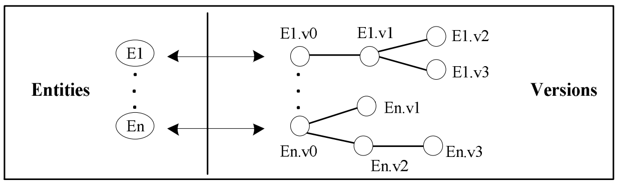


Figure 1. Derivation hierarchy

A version is either frozen or working. A frozen version describes a significant and final state of an entity. A frozen version may be deleted but not updated. To describe a new state of this entity, we have to derive a new version (from the frozen one). A working version is a version that describes one of the entity states. It may be deleted or updated to describe a next entity state. The previous state is lost to the benefit of the next one.

B. VBP2M: A meta model for versions of BPs

The VBP2M is result from merging of two layers; a BP meta model for classical BP (which not evolve on time) modeling, and versioning pattern to make some classes of the BP meta model versionable (i.e., classes for which we would like to handle versions). Because of space limitation, in this we focus on the VBP2M only. Intersected reader can consult our previous work [19] [20] to have additional information about these two layers and the way we merge them to obtain the VBP2M.

Fig. 2 below present the VBP2M in terms of classes and relationships between classes. This figure visualizes in gray versionable classes (i.e., classes for which we handle versions), and non-versionable classes (i.e., classes for which we do not handle versions). The VBP2M considers the six perspectives (Functional, operation, process, informational, organizational, and intentional perspectives).

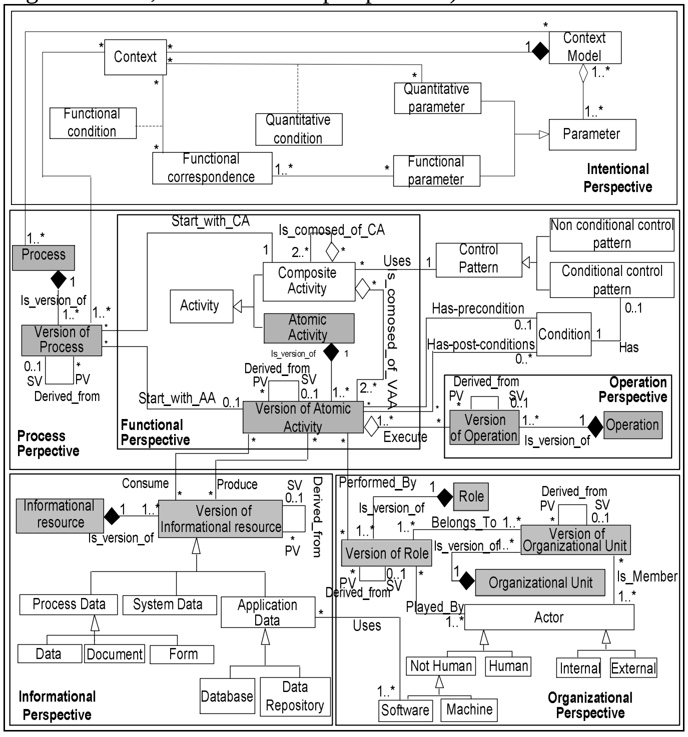


Figure 2. Versioning Business Process Meta model

1) *Main concepts of VBP2M*: The main concepts of the VBP2M are Process, Activity, Control Pattern, Operation, Informational resource, Role and Context concepts. A process performs activities, which are atomic or composite. Only the first of these activities is explicitly indicated in the meta model. At the composite activity, we keep its component activities, which are coordinated by control patterns. In our meta model, the main control patterns described in the literature are provided. Some of them are conditional (e.g., if, while, etc.), while others are not (e.g., sequence, etc.). An atomic activity can have precondition (or start condition), post-condition (or end condition) and execute one or several operations. It is performed by role, which can play by several actors belonging to organizational units (organizational perspective). Moreover, an atomic activity consumes and/or produces informational resources (informational perspective). A use context is associated for each version of process.

2) *Taking into account versions:* The underlying idea of our proposition to take into account versions of BP is to describe, for each versionable class, both entities and their

corresponding versions as indicated in “Fig. 1”. As consequence, each versionable class is described using two classes: the first class is called “…”, to model entities and a second one, called “version of …”, whose instances are versions. For instance, versions of processes are modeled within two classes: the Process class contains all modeled BP while the Version of process contains versions of the modeled BP. These classes are linked together by two relationships: the “is_version_of” relationship links a versionable class with its corresponding “Version of…” class and the “Derived_from” relationship describes version derivation hierarchies between versions of a same entity. This latter relationship is reflexive and the semantic of both sides of this relationship are: (i) a version (SV) succeeds another one in the derivation hierarchy and, (ii) a version (PV) precedes another one in the derivation hierarchy. Moreover, we introduce in the “Version of…” classes, classical properties for versions i.e., version number, creator name, creation date and state [21].

3) Versionable class: Finally, it is possible to manage versions both at the schema and the instance levels. In the Business Process context, it is only interesting to consider versions at the schema level (i.e., versions of BP schemas), and the notion of version must be applied to all the perspectives defined at the schema level. In our proposition, and unlike related work (e.g., [16] [17] [18]), which consider only two perspectives (functional and process perspectives), we take into account the five main perspectives of BPs, i.e., the process, functional, operational, organizational and informational perspectives, which are considered as relevant for BP modeling and execution [9] [10]. More precisely, regarding the process and functional perspectives, we think that it is necessary to keep versions for only two classes: the Process and the Atomic activity classes. It is indeed interesting to keep changes history for both processes and atomic activities since these changes correspond to changes in the way that business is carried out. More precisely, at the process level, versions are useful to describe the possible strategies for organizing activities while, at the activity level, versions of atomic activities describe evolution in activity execution. We defend the idea that versioning of processes and atomic activities is enough to help organizations to face the fast changing environment in which they are involved nowadays. Regarding the other perspectives, it is necessary to handle versions for the Operation class of the operational perspective, for the Informational resource class of the informational perspective, and for the Role and Organizational Unit classes of the organizational perspective.

III. MDA FRAMEWORK TO HANDEL VERSIONS OF BUSINESS PROCESSES

After modeling, verification of version of business process (VBP) can be done by (i) semantic verification and (ii) behavioral verification. The aim of the semantic verification is to verify the presence of VBP’s activities, their coordination, the invoked roles and the used informational resources. This verification can be ensured by the graphical languages and notations (i.e., BPMN, Yawl.) Regarding the behavioral verification, we verify some behavioral properties such as the liveness (i.e., the absence of global or local deadlock situation), the consistency (i.e., the existence of cyclic behavior for some marking), etc. This verification can be done with languages which have a simulator as Petri Nets. In our previous work [4], we interested by the semantic verification using BPMN. More precisely, we have generated a BPMN specification from a VBP obtained by instantiation of the VBP2M. Using this specification, we can visualize a version of a BP model in order to approve it. In this paper we deal with the behavioral verification issue. Especially, we propose mapping rules to translate automatically a VBP modeled according to VBP2M to a Petri nets specification. Then we use the Platform Independent Petri net Editor 2 "PIPE2" (which is an open source tool that contains a simulator and analyzer. It conforms to Petri Net Markup Language "PNML") to verify the behavior of the modeled VBP.

This section is organized as follows: first, we detail a MDA framework we propose to consider VBPs from modeling to execution. Second, we propose mapping rules allowing the translation from a VBP to VBP-tree. Finally, we define mapping rules to generate a Petri-net from the VBP-Tree.

A. MDA framework

An MDA (Model Driven Architecture) [22] framework specifies three levels of models: (i) the CIM (Computation Independent Model) refers to a business or domain model, (ii) the PIM (Platform Independent Model) is an independent model from all execution platforms and (iii) the PSM (Platform Specific Model) gives a textual description which can be used by execution platforms. To automate the transition from modeled VBP to the execution step, we propose an MDA framework where the CIM model contains the VBP2M and its instances (modeled VBP). The PIM model contains the specification of these VBP using a graphical specification language. At PIM model user can choose a language from (BPMN, Yawl, PN, etc.) to visualize in order to approve the modeled VBP. Regarding the PSM model an execution description (i.e., XPDL, BPEL.) is generated. This description belongs to specific platform (i.e Bonita, PETALS). This paper proposes mapping rules to translate from CIM model to PIM model. This transformation is operated according two steps: (i) consist of the querying the VBP2M in order to obtain the components elements of a VBP. These elements are organized in a tree named VBP-Tree (Versionned Business Process Tree). This step is common for all the graphical languages from PIM model (ii) allows the generation of graphical representation according to a chosen specification language using mapping rules. Fig. 3 shows the propose framework.

VBP-Tree is used to optimize the execution and the development time as the common operations will be done only once with all specification languages. For example, if we have a graphical representation of a VBP with BPMN then we want to visualize this same VBP with PN, so we just reuse the VBP-tree and just make mapping rules from VBP-Tree to BPMN.

VBP-Tree contains two types of nodes:

- Terminal node (leaves): represented by ellipses and correspond to versions of atomic activities of a VBP.
- Non terminal node: represented by rectangles and correspond to composite activities.

Fig. 4 below presents the meta model of VBP-Tree in terms of classes and relationships.

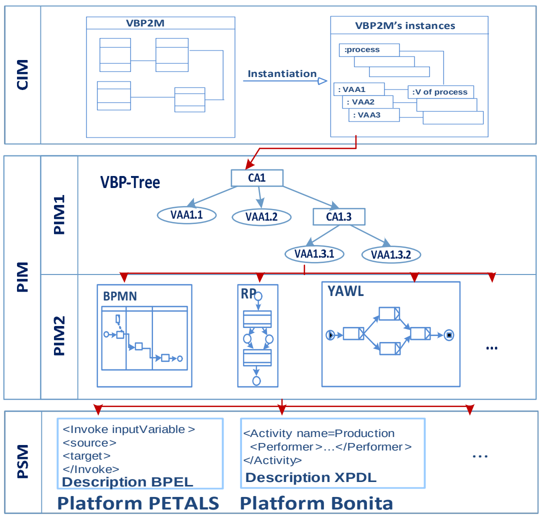


Figure 3. Framework MDA used

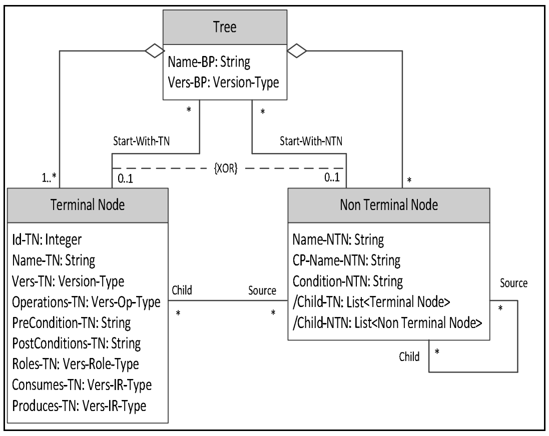


Figure 4. VBP-Tree meta model

Table I describes properties of a terminal node.

TABLE I. PROPERTIES OF TERMINAL NODE

Properties	Description
Id-TN	ID of the Terminal Node
Name-TN	Name of the Terminal node
Vers-TN	Properties of version of the atomic activity that the Terminal Node represents, it refers to another class type that contains the following properties number, Creator name, Creation Date and state of the version
Operations-TN	List of operations that are executed by the version of atomic activity that the Terminal Node represents
PreCondition-TN	Condition that must be evaluated to true to make the execution of the version of atomic activity that the Terminal Node represents
PostConditions-TN	Conditions associated to the version of atomic activity, after the execution of operations of the Terminal Node
Roles-TN	List of role able to execute the version of atomic activity that the Terminal Node represents
Consumes-TN	List of informational resources required to execute operations of the version of atomic activity that the Terminal Node represents
Produces-TN	List of informational resources produced after executing operations of the version of atomic activity that the Terminal Node represents

Properties of a non terminal node are described in Table II:

TABLE II. PROPERTIES OF NON TERMINAL NODE

Properties	Description
Name-NTN	Name of the Non Terminal Node
CP-Name-NTN	Name of the control pattern used for the composite activity that the Non Terminal Node represents
Condition-NTN	Optional property associated to conditional control patterns
Child-TN	List of Terminal Node that compose the Non Terminal Node
Child-NTN	List of Non Terminal Node that compose the Non Terminal Node

In the remainder, we represent firstly mapping rules from VBP2M to VBP-Tree. Secondly, we generate a PN with mapping rules from VBP-Tree to PN.

B. Mapping rules from VBP2M to VBP-Tree

To translate from a VBP to a VBP-Tree, we propose three mapping rules detailed in Table III below.

TABLE III. MAPPING RULES FROM VBP2M TO VBP-TREE

N°	VBP2M concepts	VBP-Tree concepts
1	Version of Business Process	Tree
2	Version of Atomic Activity	Terminal Node
3	Composite Activity	Non Terminal Node

The function implementing the mapping from a VBP to VBP-Tree (cf. Fig. 5) use a set of functions permitting the handling version of processes and nodes

- StartWithVAA (VP): indicates if the VP start with Version of Atomic Activity class;
- BuildTN (A, Tree): build a Terminal Node with properties of the Version of Atomic Activity (A) and its relations (performed-by, consumes, etc.), then add it in the Tree;

- BuildNTN (A, Tree): build a Non Terminal Node with properties of the composite Activity and its relations (uses, etc.), then add it in the Tree;
- Children (Node): return all the children of the composite activity (Node).

```
Function BuildVBP-Tree (A: Activity): VBP-Tree
Local n: Node
Global Tree: VBP-Tree
Begin
   If StartWithVAA (A)
      n: BuildTN (A, Tree)
   Else
      n: BuildNTN (A, Tree)
      For each child in Children (A)
         BuildVBP-Tree = BuildVBP-Tree (child)
      Next child
   End If
End.
```

Figure 5. Function BuildVBP-Tree

C. *Mapping rules from VBP-Tree to PN*

Firstly, we explain more what is PN? In fact, Petri Net is a formal tool that is composed of:

- A set of places (P1, P2, ..., Pn) which represents triggering conditions;
- A set of transitions (T1, T2, ..., Tn) which represent activities;
- A set of oriented arcs. Two types of arcs are distinguished: input arcs which link place to transition and output arcs which link transition to place. For the input arcs, we distinguish also two types which are normal (arc with an arrow at the end, that can contains a token) and inhibitor (arc with a small circle at the end, that cannot contains a token) arcs.

The meta model of PN is shown in the UML diagram of Fig. 6.

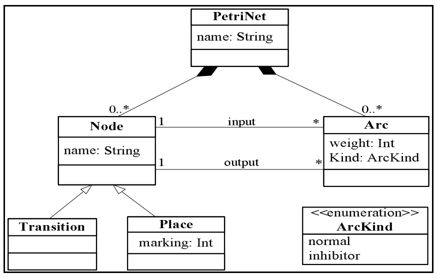


Figure 6. Petri net meta model

In order to support concepts of the VBP2M, we propose to extend Petri net meta model by adding five classes:

- Version class: that contains version number, creator name, creation date and state attributes. This class linked by the Node class and the Petri Net class in order to specify their versions;
- Operation class: contains an attribute which specify its name and has a relationship named "op-vers" that is linked to Version class;
- Informational resource class: specify the type of the Place class. It contains two attributes: type and nature of the resource;
- Role class: specify the type of the Place class;
- Organizational unit class: specify the type of the Place class;

Besides these new classes, we add also two attributes in the Transition class that concerns the precondition and the post conditions and one relationship named "has-vers-op" linked to Version class. The aim of this extension is to increase the semantic dimension by specifying each node with its version and non version information. Fig. 7 shows the extended meta model of PN. These new classes are represented with gray color.

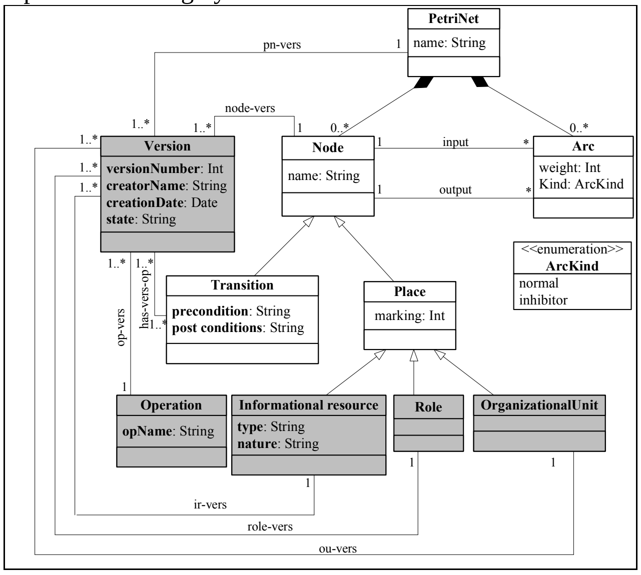


Figure 7. Petri net meta model extended

To generate a PN from VBP-Tree we propose mapping rules (cf. table IV) between VBP-Tree properties and the extended PN properties. After applying these rules, it becomes possible to verify the behavioral of the obtained PN. This verification ensured using the simulator of PIPE2.

TABLE IV. MAPPING RULES FROM VBP-TREE TO PN

VBP-Tree concepts	**VBP-Tree properties**	**PN properties**
Terminal Node	Name-TN	Name of a transition
	Name-Role of the attribute Roles-TN	Name of an input Role or Organizational unit place drawn with an inhibitor arc
	Name-IR of the attribute Consumes-TN, when Type-IR is “internal” or “external”	Name of an input Informational resource place drawn with a normal arc
	Name-IR of the attribute Consumes-TN, when Type-IR is “position”	Name of an input Informational resource place drawn with an inhibitor arc
	Name-IR of the attribute	Name of an output

	Produces-TN	Informational resource place drawn with a normal arc
Tree	Name-BP and Vers-BP	Name of the Petri net

We use the inhibitor arc for resources (Role and/or Information) that are not consumed by an atomic activity. These mapping rules are implemented by the function “Build PN” detailed in Fig. 8:

```
Function BuildPN (n: Node): PN
Local child: Node
Begin
  If IsTN(n)
    t: BuildTransitionPlace (n)
  Else
    Case pattern (n)
      When sequence: t: BuildTranPlacSeq(n)
      When parallel: t:BuildTranPlacPar(n)
      When …
    End case
    For each child in children(n)
      BuildPN(child)
    Next child
  End If
End.
```

Figure 8. Function BuildPN

This function uses a set of functions:

- IsTN(n): indicates if a node n is a terminal node;
- BuildTransitionPlace (n): build and add the corresponding transition and places of the terminal node n;
- *Pattern* (n): return the used pattern if n is a non terminal node;
- *BuildTranPlacSeq*(n) : build and add the transition and places of the terminal node n into the PN according to a sequence control pattern;
- *BuildTranPlacPar* (n): build and add the transition and places of the terminal node n into the PN according to a parallel control pattern.

Because of space limitation, we do not specify other control pattern such choice, iteration, etc.

IV. CASE STUDY

In order to illustrate our approach, we propose the process of the participation in a business tender named BTP .

The first version of this process, represented in Fig. 9(a), contains three activities (Acquisition of tender specifications, Preparation of the offer, Submission of the offer).

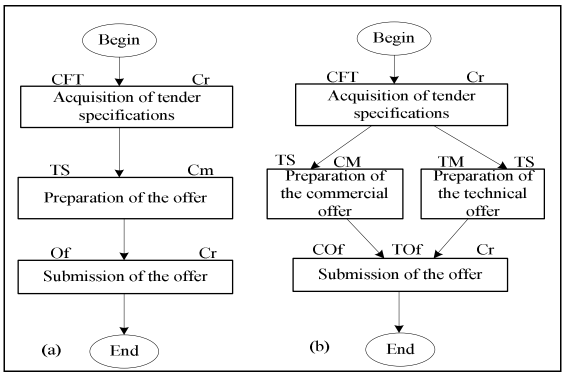

Begin
CFT
Cr
Acquisition of tender specifications
TS
Cm
Preparation of the offer
Of
Cr
Submission of the offer
(a)
End
Begin
CFT
Cr
Acquisition of tender specifications
TS
CM
TM
TS
Preparation of the commercial offer
Preparation of the technical offer
COf
TOf
Cr
Submission of the offer
(b)
End

Figure 9. Versions of process

- “*Acquisition of tender specifications*” which is triggered by the presence of a call for tender (CFT) and produces a tender specifications (TS). This activity is achieved by a courser (Cr).
- “*Preparation of the offer*” which is triggered by the availability of specifications tender and produces an offer (of). This activity is done by a committee of offer preparation (Cm).
- “*Submission of the offer*” which is triggered by the prepared offer and produces a coupon. This activity is realized by the courser.

Fig. 10, gives an extract of an instantiation of the VBP2M according to the first version of BTP process.

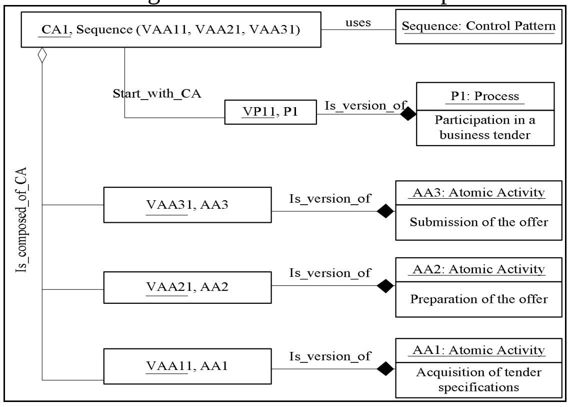


Figure 10. Instantiation of the VBP2M for the first version of BTP process

Fig. 11 presents the VBP-Tree of this version with a simplified view (nodes are not described in details).

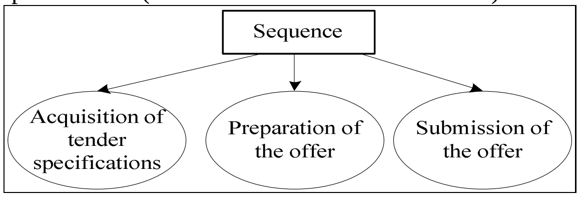


Figure 11. VBP-Tree for the first version of the BTP process

In the second version of this process, represented in Figure 9(b), the second activity “*Preparation of the offer*” will be divided: (i) “*Preparation of the technical offer*” activity which is realized by a technical manager (TM) (ii)

"*Preparation of the commercial offer*" activity which is realized by a commercial manager (CM). These two activities are done in parallel and executed respectively by a *Technical service* and *Commercial service*. The VBP-Tree of this version is illustrated in Fig. 12

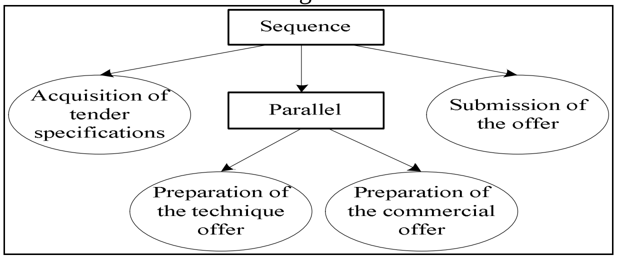

Sequence
Acquisition of tender specifications
Parallel
Submission of the offer
Preparation of the technique offer
Preparation of the commercial offer

Figure 12. VBP-Tree for the second version of BTP process

V. IMPLEMENTATION

We use PIPE2 (Platform Independent Petri net Editor 2) [23] [24] tool to implement our propositions.

Two steps are considered to implement our propositions: (i) extend the Petri net tool "PIPE2" in order to augment the existing dialogues that open when you currently click on the Place or Transition objects to display proprieties relative to our context and (ii) implement the proposed mapping rules.

In fact, we add new package "vffs" that contain:

- Four forms: One associate to the Transition class and the three others associate to the Place class (Role Place, Informational resource Place and Organizational unit Place);
- A class: contains methods that fill these forms.

We modify also:

- The "PlaceHandler.java" and "TransitionHandler.java" classes in the package "pipe.gui", especially the method of "mouseclicked";
- The "Place.java" and "Transition.java" in the package "pipe.dataLayer", especially the method of "showEditor".

After extending PIPE2, we implement mapping rules of detailed in sections III.B and III.C. Fig. 13 shows further the steps to generate automatically PN from the VBP2M, passing through the VBP-Tree.

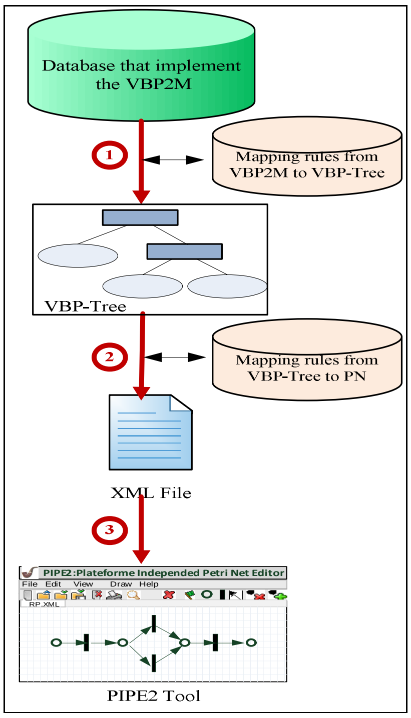
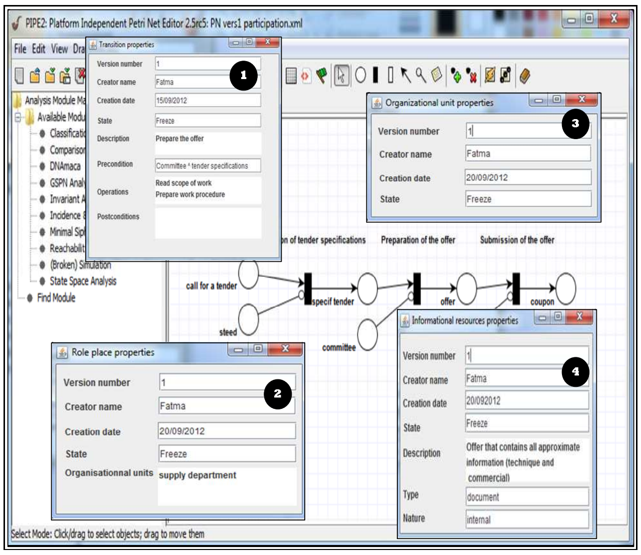

Database that implement the VBP2M
1
Mapping rules from VBP2M to VBP-Tree
VBP-Tree
2
Mapping rules from VBP-Tree to PN
XML File
3
PIPE2:Plateforme Independed Petri Net Editor
File Edit View Draw Help
RP.XML
PIPE2 Tool

Figure 13. Steps of implementation

In fact, these steps are:

- 1: This step allows to translate from VBP2M to VBP-Tree according the mapping rules detailed in § III.B.
- 2: As PIPE2 save/load its Petri nets in XML file, we generate this file from the VBP-Tree according the mapping rule explained in § III.C. To create the XML file we used the JDOM API (which enables to parses, manipulates, and outputs XML using standard Java constructs).
- 3: Finally, we open the created XML file by PIPE2 to verify by simulation the nets that belongs to a specific VBP.

After choosing a VBP and building the VBP-Tree, we choose PN to visualize and simulate our VBP. The result is shown in Fig. 14.


PIPE2: Platform Independent Petri Net Editor 2.5rc5: PN vers1 participation.xml

Transition properties (1)

Version number	1
Creator name	Fatma
Creation date	15/09/2012
State	Freeze
Description	Prepare the offer
Precondition	Committee ^ tender specifications
Operations	Read scope of work Prepare work procedure
Postconditions	

Role place properties (2)

Version number	1
Creator name	Fatma
Creation date	20/09/2012
State	Freeze
Organisationnal units	supply department

Organizational unit properties (3)

Version number	1
Creator name	Fatma
Creation date	20/09/2012
State	Freeze

Informational resources properties (4)

Version number	1
Creator name	Fatma
Creation date	20/092012
State	Freeze
Description	Offer that contains all approximate information (technique and commercial)
Type	document
Nature	internal

Select Mode: Click/drag to select objects; drag to move them

Figure 14. The generation of the first version of participation in a business tender process

- When we click in a transition we obtain on (1) a form displaying its information such version number, creator name, creation date, state, description, precondition, operations and post conditions.
- When we click in a role place we obtain on (2) a form displaying its version information such version number, creator name, creation date, state and organizational units that the role place belongs.
- When we click in an organizational unit place we obtain on (3) a form displaying its version information.
- When we click in an informational resource place we obtain on (4) a form displaying its information such version number, creator name, creation date, state, type and nature.

Fig. 15 shows the simulation result of this VBP visualized in Fig. 14. In fact, we can conclude that all transitions are attainable. So, the reachability property is verified. There are many other properties that can be verified such as the liveness property, the boundedness property, etc. We can also use the analysis techniques that the PIPE2 tool provides.

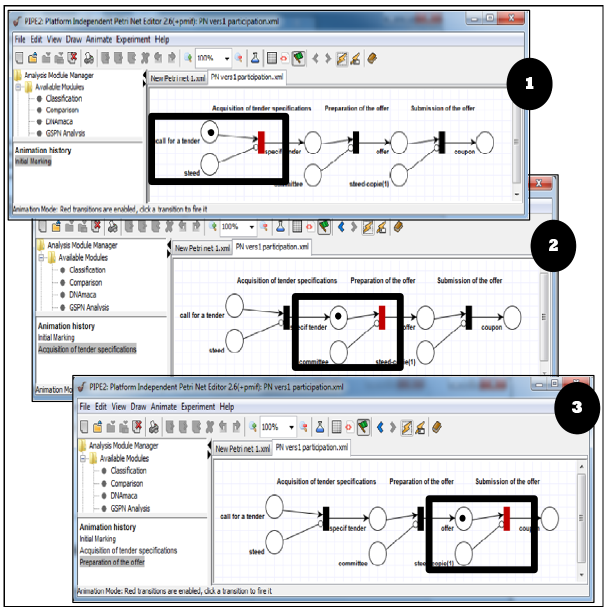

PIPE2: Platform Independent Petri Net Editor 2.6(+pmif): PN vers1 participation.xml
File Edit View Draw Animate Experiment Help
Analysis Module Manager
Available Modules
Classification
Comparison
DNAmaca
GSPN Analysis
Animation history
Initial Marking
Acquisition of tender specifications
Preparation of the offer
Submission of the offer
call for a tender
specif tender
offer
coupon
steed
committee
steed-copie(1)
Animation Mode: Red transitions are enabled, click a transition to fire it
1
2
3

Figure 15. The simulation result of the first version of participation in a business tender process

VI. CONCLUSION AND FUTURE WORK

This paper has presented a solution to simulate the behavioral dimension of versions of BP using an extension of PN. This solution is integrated into a more general framework supporting a process designer for modeling and specifying flexible business processes using the version concept. This framework advocates a MDA approach considering (i) at the CIM level, a specific meta model, the Version Business Process meta model (VBP2M) for modeling versions of BPs (ii) at the PIM level, an extension of the PN meta model for validating by simulation the behavioral dimension of modeled BP versions, and finally, (iii) at the PSM level, several meta models for implementing versions of BPs (e.g., XPDL and BPEL meta models). This paper mainly focuses on the automatic mapping from the CIM level onto the PIM level (i.e., the extension of the PN meta model). Its contributions are the following:

- The specification of an extension of PN in order to support the versions of BPs.
- An automatic mapping of versions of BP modeled according the VBP2M onto BP versions modeled with extended PN meta model.

An implementation of this mapping extending the PIPE2 tool in order to take into account version specificities. Regarding related work, main contributions in BPs [16][17][18] only considered two perspectives (functional and process) and did not consider four other perspectives (operation, informational, organizational and intentional) which are considered as relevant for BP [19][20]. Moreover, theses contributions do not address the mapping from the modeled versions to their graphical representation.

Our future work will take two directions. First perspective, an extensive study of control pattern in PN because we only considered sequence, parallelism and choice patterns. Other more long perspective, we will map versions of BP modeled using the extended PN meta model onto versions of BP described using language relevant from the PSM level of our MDA-based framework: XPDL and BPEL, which are the de-facto standards for implementing BP. Second, we will address execution of specified BP.

REFERENCES

[1] H. Smith and P. Fingar, "Business Process Management: the Third Wave business process modelling language (bpml) and its pi-calculus foundations," in the Information and Software Technology 45(15), 2003, pp. 1065-1069.

[2] M. Dumas, W. van der Aalst and A. ter Hofstede, "Process-Aware Information Systems: Bridging People and Software through Process Technology," Wiley-Interscience, 2005.

[3] W. van der Aalst, B. Benatallah, F. Casati, F. Curbera and E. Verberk, "Business Process Management: Where Business Processes and Web Services Meet," in the Int. Journal on Data and Knowledge Engineering, 61(1), 2007, pp. 1–5.

[4] I. Ben said, M.A. Chaâbane and E. Andonoff, "A Model Driven Engineering Approach for Modelling Versions of Business Processes using BPMN," in the Int. Conference on Business Information System (BIS'10), Berlin, Germany, May 2010, pp. 254–267.

[5] F. Casati, S. Ceri, B. Pernici and G. Pozzi. "Workflow Evolution," in the Int. journal of Data and Knowledge Engineering. 24(3), 1998, pp. 211–238.

[6] G. Faustmann, "Enforcement vs. Freedom of Action - An Integrated Approach to Flexible Workflow Enactment," in the ACM SIGGROUP Bulletin, 20(3),1999, pp. 5–6.

[7] P. Kammer, G. Bolcer, R. Taylor and M. Bergman, "Techniques for supporting Dynamic and Adaptive Workflow," in the Int. Journal on Computer Supported Cooperative Work, 9(3/4), 2000, pp. 269–292.

[8] S. Rinderle, M. Reichert and P. Dadam, "Disjoint and Overlapping Process Changes: Challenges, Solutions and Applications," in the Int. Conference on Cooperative Information Systems, Agia Napa, Cyprus, 2004, pp.101–120.

[9] W. van der Aalst, A. ter Hofstede, B. Kiepuszewski and A. Barros, "Workflow Patterns," in the Int. Journal on Distributed and Parallel Databases, 2003, 14(1), pp. 5–51.

[10] S. Nurcan, "A Survey on the Flexibility Requirements related to Business Process and Modelling Artifacts," in the Int. Conference on System Sciences, Waikoloa, Big Island, Hawaii, USA, January 2008, pp. 378–387.

[11] H. Schoneneberg, R. Mans, N. Russell, N. Mulyar and W. van der Aalst, "Process Flexibility: A Survey of Contemporary Approaches," in the Int. Workshop on CIAO/EOMAS, at Int. Conf. on Advanced Information Systems, Montpellier, France, June 2008, pp. 16–30.

[12] M. A. Chaâbane, E. Andonoff, L. Bouzguenda and R. Bouaziz. "Versions to Address Business Process Flexibility Issue," in the East-European Conference on Advances in Databases and Information Systems (ADBIS 2009), Riga, 07/10/2009 – 10/10/2009, Janis Grundspenkis, Tadeusz Morzy, Gottfried Vossen (Eds.), Springer Berlin, Heidelberg, September 2009, pp. 2–14.

[13] M. Pesic, H. Schonenberg and W. van der Aalst, "Constraint-Based Workflow Models: Change Made Easy," in the Int. Conference on Cooperative Information Systems, Vilamoura, Portugal, November 2007, pp. 77–94.

[14] W. van der Aalst "Three Good reasons for Using a Petri-net-based Workflow Management System, " In proceedings of the International Working Conference on Information and Process Integration in Entreprises (IPIE'96), Cambridge, Massachusetts, USA, November 1996, pp. 179 – 201.

[15] S. Narayanan and S. McIlraith, "Simulation, Verification and Automated Composition of Web Services, " in the 11th Int. World Wild Web Conference, Honolulu, Hawaii, 2002, pp. 77–88.

[16] M. Kradofler and A. Geppert, "Dynamic Workflow Schema Evolution based on Workflow Type Versioning and Workflow Migration," in the Int. Conference on Cooperative Information Systems, Edinburgh, Scotland, 1999, pp. 104–114.

[17] X. Zhao and C. Liu, "Version Management in the Business Change Context," in the Int. Conference Business Process Management, Brisbane, Australia, September 2007, pp. 198–213.

[18] B. Weber, M. Reichert and S. Rinderle-Ma, "Change Patterns and Change Support Features -Enhancing Flexibility in Process-Aware Information Systems," in Data and Knowledge Engineering 66 (3), September 2008, pp. 438–466.

[19] M. A. Chaâbane, E. Andonoff, L. Bouzguenda and R. Bouaziz, "Dealing with Business Process Evolution," in the Int. Conference on E-Business (ICE-B 2008), Porto, Portugal, July 2008, pp. 267–278.

[20] M. A. Chaâbane, E. Andonoff, R. Bouaziz and L. Bouzguenda, "Modélisation Multidimensionnelle des Versions de Processus," Ingénierie des Systèmes d'Information, Numéro spécial Modélisation d'entreprise, RTSI ISI 15(5), 2010, DOI:10.3166, Lavoisier, Paris, pp. 89–114.

[21] E. Sciore. "Versioning and Configuration Management in Object-Oriented Databases," in the Int. Journal on Very Large Databases, 3(1), 1994, pp. 77–106.

[22] OMG, MDA Guide Version 1.0.1, Document Number: omg/2003-06-01. OMG, Juin 2003.

[23] PIPE Homepage. http://pipe2.sourceforge.net/

[24] P. Bonet, C. M. Llado, R. Puigjaner and W. J. Knottenbelt, "PIPE v2.5: a Petri Net Tool for Performance Modeling," in 23rd Latin American Conference on Informatics, October 2007.