BPMN4V
An Extension of BPMN for Modelling Adaptive Processes using Versions

Imen Ben Said[1], Mohamed Amine Chaâbane[1], Eric Andonoff[2] and Rafik Bouaziz[1]
[1]*University of Sfax, MIRACL, route de l'aéroport, BP 1088, 3018 Sfax, Tunisia*
[2]*UT1-IRIT, 2 rue du Doyen Gabriel Marty, 31042 Toulouse Cedex, France*

Keywords: BPMN4V, BPMN4V-Modeller, Versions, BPMN, Process Adaptation, Process Variability.

Abstract: This paper presents BPMN4V, an extension of BPMN 2.0 to support business process adaptation modelling using versions. It introduces the provided extensions to the BPMN meta-model to take into account the notion of version, considering both static and dynamic aspects of process versions. It also presents BPMN4V-Modeller, an implementation of these extensions. Therefore, using BPMN4V business process designers can model process adaptation, which is an important issue to address before the definitive acceptance and use of business process management systems in companies.

1 INTRODUCTION

Versions are now known to be a powerful mechanism to address business process adaptation (Kradolfer and Geppert, 1999); (Zhao and Liu 2007), which is a key challenge for the definitive acceptance and use of process information systems in companies (Smith and Fingar, 2003); (Dumas et al., 2005); (Weske, 2007). First, versions are useful to facilitate the migration of instances from an old process schema to a new one. Indeed, changes performed on running instances may affect activities already executed, making the migration impossible (Casati et al., 1996): then versioning is very useful as it allows the running of several instances of a same process according to different schemas. In addition, as defended in (Chaâbane et al., 2009), versions are appropriate to address the three main types of process adaptation (Nurcan, 2008); (Schonenberg et al., 2008); (Weber et al., 2009): (i) *adaptation by design*, for handling foreseen changes in processes where strategies to face these changes are not necessarily defined at design-time but must be specified at run-time (*e.g.,* late modelling and late binding (Adams et al., 2006)), (ii) a*daptation by deviation*, for handling occasional unforeseen changes and where the differences with the initial process are minimal, and, (iii) *adaptation by evolution*, for handling unforeseen changes in processes, which require occasional or permanent modifications in process schemas. More precisely, when modelling versions of process schemas, it is possible to model both process schema evolution and process schema variability (using alternative versions). Consequently, versions enable business process designers to address adaptation by evolution, adaptation by design, including late modelling and late binding, and also adaptation by deviation (Chaâbane et al., 2009).

We examined different contributions from literature which advocate the introduction of the version notion to deal with process adaptation (*e.g.*, (Kradolfer and Geppert, 1999); (Zhao and Liu, 2007); (Chaâbane et al., 2009); (Lu and Sadiq, 2009); (Hallerbach et al., 2010). These propositions are interesting but they have two main drawbacks. First, they mainly focus on the behavioural dimension of processes, leaving aside their organizational and informational dimensions. However, these dimensions have also to be considered when dealing with process adaptation, since adaptation may be related to the resources involved during process execution or to the information being managed during process execution. Secondly, each of these contributions introduce specific notations, which are not standards and are unlikely to be used by process designers who are in charge of modelling variability of processes.

On the other side, BPMN is recognized as the de-facto standard for business process modelling. Promoted by the OMG (OMG, 2011), it is widely

Ben Said I., Amine Chaâbane M., Andonoff E. and Bouaziz R..
BPMN4V - An Extension of BPMN for Modelling Adaptive Processes using Versions.
DOI: 10.5220/0005376102580267
In *Proceedings of the 17th International Conference on Enterprise Information Systems* (ICEIS-2015), pages 258-267
ISBN: 978-989-758-098-7

used by process designers involved in business process projects to model processes as a set of synchronized activities and events. However, BPMN does not integrate concepts to model variability of processes.

Some extensions have been recommended to face this drawback. First, (Korherr and List, 2007) and (Santos et al., 2010) extended BPMN with the notion of goal to deal with process adaptation issue. For instance, in (Korherr and List, 2007), goal models are used to define the possible variations of processes. However several different process schemas may co-exist all together while sharing the same goal. Therefore, only considering goal is not enough to deal with adaptation by design, adaptation by deviation and adaptation by evolution. Secondly, (Ben Said et al., 2010) and (Ben Said et al., 2014a) recommended the extension of BPMN with the notion of version to address process adaptation issue. On the one hand, (Ben Said et al., 2010) advocated the use of the VBP2M meta-model to define versions of processes, and recommended a model driven engineering approach to have a BPMN graphical representation of process versions modelled as instances of VBP2M. However, this contribution does not really extend BPMN, which is only used as a target notation for process version modelling. Moreover, the proposed meta-model does not integrate all the BPMN concepts and needs to be improved to provide a more comprehensive view of the modelled processes. On the other hand (Ben Said et al., 2014a) really extended the BPMN meta-model for process version modelling. However (Ben Said et al., 2014a) has the following drawbacks: (i) it does not introduce any graphical notation for process version modelling, only focusing on the introduction of the notion of version in BPMN meta-models, (ii) it does not investigate the dynamic aspects of process version management, and finally, (iii) it does not present any tool supporting process version modelling.

This paper extends the proposition of (Ben Said et al., 2014a) addressing the dynamic aspects of process version management defining several states for process versions and corresponding operations. It also recommends a graphical notation for representing version concepts. This notation is used in BPMN4V-Modeller, a tool intended to business process designers, and supporting the modelling of BPMN business process versions. Note that this paper focuses on intra-organizational processes, which are modelled in BPMN through private processes, which are internal to a specific organization. It does not consider inter-organizational processes (modelled as collaborations in BPMN) which will be the focus of a future work.

This paper is organized as follows. Section 2 presents the BPMN4V meta-model which extends BPMN 2.0 meta-model to deal with process adaptation/variability using versions. Section 3 focuses on the dynamic aspects of BPMN4V; it introduces the different states a version may have during its life cycle, along with the basic operations for version management. Section 4 presents BPMN4V-Modeller, a specific tool for process version modelling along with the graphical notation advocated for the notion of version. Finally, section 5 concludes the paper and gives some directions for future works.

2 BACKGROUND: THE BPMN4V META-MODEL

Figure 1 gives an overview of the BPMN4V meta-model (Ben Said et al., 2014a), which combines BPMN 2.0 concepts for the modelling of the three main dimensions of processes, *i.e.* the behavioural, the informational and the organizational dimensions, along with the notion of version.

More precisely, the behavioural dimension of a process supports the description of the process activities and their synchronization along with the events happening during its execution through the notion of *FlowElementContainer* which gathers *SequenceFlow*, *FlowNode* (*Gateway*, *Event*, and *Activity*), and *Data Object*. A *SequenceFlow* is used to show the order of *FlowNode* in a process. It may refer to an *Expression* that acts as a gating condition. A *Gateway* is used to control how *SequenceFlow* interact within a process. An *Event* is something that happens during the course of a process. It can correspond to a trigger, which means that it reacts to something (*catchEvent*), or it can throw a result (*throwEvent*). An *Event* can be defined by one or more *EventDefinitions*. An *Activity* is a work performed within a process. An *Activity* can be a *Task* (*i.e.*, an atomic activity) or a *Sub Process* (*i.e.*, a non-atomic activity). A *Task* is used when the work is elementary (*i.e.*, it cannot be more refined).

Regarding the organizational dimension of processes, an activity is performed by a *ResourceRole*. A *ResourceRole* can refer to a *Resource*. A *Resource* can define a set of parameters called *ResourceParameters*. A *ResourceRole* can be a *Performer*, which can be a *HumanPerformer*, which can be in turn a *PotentialOwner*.

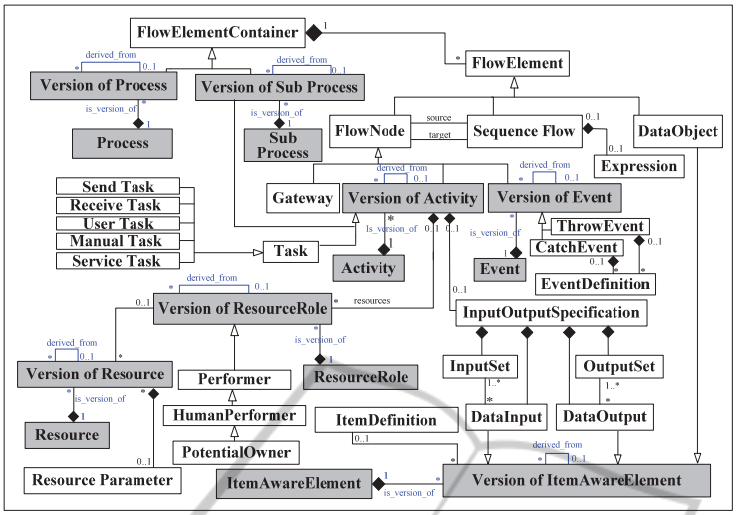

Figure 1: BPMN4V Meta-model for Modelling Process Version.

Regarding the informational dimension of processes, an *ItemAwareElement* references element used to model the items (physical or information items) that are created, manipulated and used during a process execution. An ItemAwareElement can be a DataObject, a DataObjectReference, a Property, a DataStore, a DataInput or a DataOutput.

In order to take into account the notion of version in BPMN 2.0, (Ben Said et al., 2014a) recommends modelling for each *versionable* concept both the concept itself and the versions it gathers. For instance, (Ben Said et al., 2014a) advocates to model the processes themselves in a class and their versions in a separate class. Two specific relationships are added between these two classes: the *is_version_of* relationship which makes a link between a concept and its versions, and the *derived_from* relationship which makes a link between the versions themselves.

(Ben Said et al., 2014a) recommends to handle versions for seven BPMN 2.0 concepts in order to model process variability: *Process*, *Sub Process*, *Event*, *Activity*, *ItemAwareElement, Resource*, and *ResourceRole*. The idea is to keep track of changes occurring to components participating to the description of the way business is carried out. Therefore, as explained before, for each of these concepts, BPMN4V provides two classes and two relationships, shown in grey and blue in Figure 1.

3 DYNAMIC ASPECTS OF BPMN4V

In order to handle versions of processes modelled as instances of the BPMN4V meta-model, we recommend a taxonomy of operations which allows creating, deriving, updating, validating and deleting process versions. This section introduces these operations, giving a state chart indicating when they are available, and detailing the actions they perform according to the classes in which they are defined.

3.1 State Chart for Versions

The UML state chart given in Figure 2 indicates when operations for versions are available with respect to the version state. Some of them are available whatever the state of the version on which they are performed, while others are available only in some cases. In the state chart, each operation is described using the notation Event/Operation whose meaning is “if Event appears then Operation is triggered”.

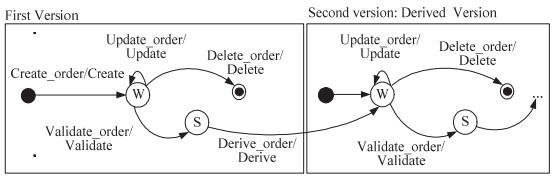

Figure 2: UML State Chart for Versions.

When the *Create order* event appears, the *Create* operation is carried out to create both a concept (*e.g.*, a process) and its corresponding first version. The state of the created version is *working* (W). In this state, a version is not yet a final one, and it can be updated (*Update* operation). It can also be deleted (*Delete* operation) or validated (*Validate* operation). When the *Validate* operation is performed, the

corresponding version becomes *stable* (S). This state indicates that a version is a final one, on which no additional updates can be performed. A stable version can serve as basis for the creation of a new version using the *Derive* operation. The created version has the same value as the version from which it is derived, and its state is *working*.

3.2 Operations for Version Management

In addition to the previous state chart, these operations require further details. For instance, *Create* or *Update* operations permit to define (add or delete) elements of the version components. These components change according to the considered type of version on which the operation is performed: version of process, version of activity (sub-process or task), version of event, version of ItemAware Element, version of Resource and version of ResourceRole. Moreover, regarding the *Derive* and *Validate* operations, they can trigger the derivation or the validation of versions of their components. Sections below give the semantics of these operations.

3.2.1 Create and Update Operations

Table 1 gives the definition of Create and Update, indicating the actions these operations include.

Table1: Create and Update Operations.

VBPMN Concepts	Actions
Process/Sub-Process	+/- Activities +/- Events +/- Sequence Flows +/- Gateways
Activity	+/- Information +/- Resource Role Define Type
Event	Define Type Define Information
ItemAwareElement	Define Type Define Structure
Resource	+/- Parameter
ResourceRole	Define Resource Define Type

For instance, the creation or the update of a process or a sub-process includes actions supporting the addition (+) or the deletion (-) of activities, events, sequence flows and gateways to define the behavioural dimension of the process or the sub-process. In the same way, the creation or the update of an activity includes actions supporting the addition (+) or the deletion (-) of ItemAware Elements produced or required by the activity, the addition (+) or the deletion (-) of ResourceRole (*e.g.*, to define the role involved in activity realization), and the definition of the activity type.

Therefore, Table 1 indicates that the *Create* and *Update* operations change according to the classes in which they are defined. However, they share the same general idea that is to give values to properties and relationships of the considered classes.

3.2.2 Validate Operation

The *Validate* operation is performed if a working version becomes stable (*i.e.*, it does not need additional updates). Validation of a version may trigger the validation of other versions, which are linked to it. Figure 3 shows the validation propagation.

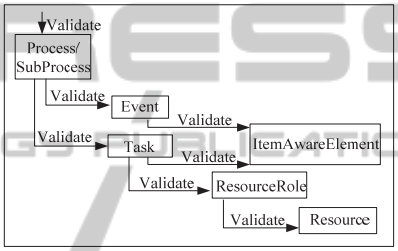

Figure 3: Validation propagation.

For instance, the validation of a process version triggers the validation of its versioned components (*i.e.*, versions of activities and versions of events). In the same way, the validation of an activity triggers the validation of versions of ItemAwareElement (data) and versions of ResourceRole it references.

3.2.3 Derive Operation

The *Derive* operation allows creating a new version from an existing stable version. The created version is a working version. Before being updated, it has the same value as the derived one. Moreover, derivation of a version may trigger the derivation of other versions, which are linked to the derived one. Figure 4 illustrates this derivation propagation.

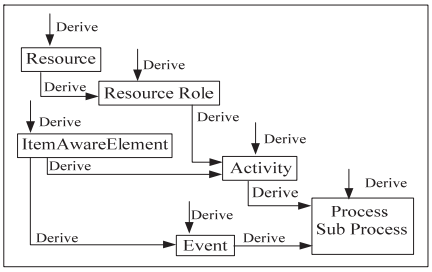

Figure 4: Derivation propagation.

This propagation is due to the composition relationships existing between Process (FlowElemen tContainer) and Activity, Event, ItemAwareElement, Resource and ResourceRole (FlowElement). Therefore, the derivation of a resource triggers the derivation of its corresponding ResourceRole and the derivation of a ResourceRole or an ItemAwareElement triggers the derivation of the corresponding activity. In the same way, the derivation of an activity or an event triggers the derivation of its corresponding process or sub-process.

4 BPMN4V-MODELLER

BPMN4V-Modeller is a dedicated tool for modelling business process versions as instances of the BPMN4V meta-model. It is an extension of the already existing Eclipse BPMN Modeller plug-in: it extends this later by integrating (i) new icons for representing versions, (ii) new Eclipse views to show version details and, (iii) new contextual menus and Eclipse dialogs to implement dynamic aspects of versions.

This section illustrates the use of BPMN4V Modeller through an example of adaptive process. More precisely, the section introduces the *Damage Compensation* process example, gives a brief overview of the BPMN4V-Modeller and illustrates how we can use it to model versions of the Damage Compensation process.

4.1 Illustrative Example

Figure 5 hereafter shows two versions of the Damage Compensation process of an insurance company. Basically, this process contains five activities: *Receive Request*, *Review Request*, *Send reject letter*, *Calculate claim amount* and *Financial settlement*. The first version is given in Figure 5(a). This version starts when a client files a claim. After checking the claim, a reject letter is sent if the request is not accepted. Otherwise, the claim amount is calculated by the insurance manager using *Grid Calculator*, and the financial service prepares and sends the financial settlement. In addition, the insurance agency has modelled a second version of this process to face an increasing number of its customers. Figure 5(b) shows this new version, introducing an *Expertise* activity (a new activity used when the damage amount exceeds 1000$) and modifying both the start *Claim Filed* event and the *Receive Request* and *Calculate claim amount* activities (their type have changed). To sum up, we have two versions of the *Damage Compensation* process, two versions of the *Claim Filed* event and two versions of the *Receive Request* and *Calculate claim amount* activities: the first version of both *Receive Request* and *Calculate claim amount* activity are used in the first version of the process while the second ones are used the second version of the process. In addition, the sequence flows and gateways have been modified in the second version of the process. Finally, regarding the *Claim Filed* event, in the first version of the process, it is a none start event indicating that this version starts when the client gives its *Claim File* (a paper data); in the second process version, it becomes a message event, indicating that the client sends the *Claim File* (an electronic data) as a message via the insurance web site.

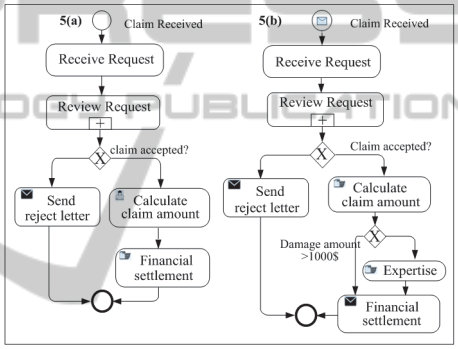

Figure 5: Versions of the damage compensation process.

4.2 BPMN4V Modeller Overview

Figure 6 gives an overview of BPMN4V-Modeller. The central part of the screenshot (part ❶) is the drawing canvas of the process version schemas. The right part of the figure (part ❷) defines the set of widgets that can be used to define process version schemas: events, activities, gateways and sequence flows. The left part of the figure (part ❸, ❹ and ❺) corresponds to the added Eclipse views.

More precisely, the drawing canvas provides multiple tabs, each one being used to model and display a separate BPMN diagram. Each diagram represents a particular version of a process. To highlight the notion of version, we decorate the tab name, the activity shape and the event shape with the following icon . This icon is used to mark that any element (process, activity or event) shown in the drawing canvas is a version. For instance, in Figure 6, the drawing canvas contains the process *VP1-1* (the icon decorates the VP1-1 tab); in addition, *VP1-1* contains two versions of events and five

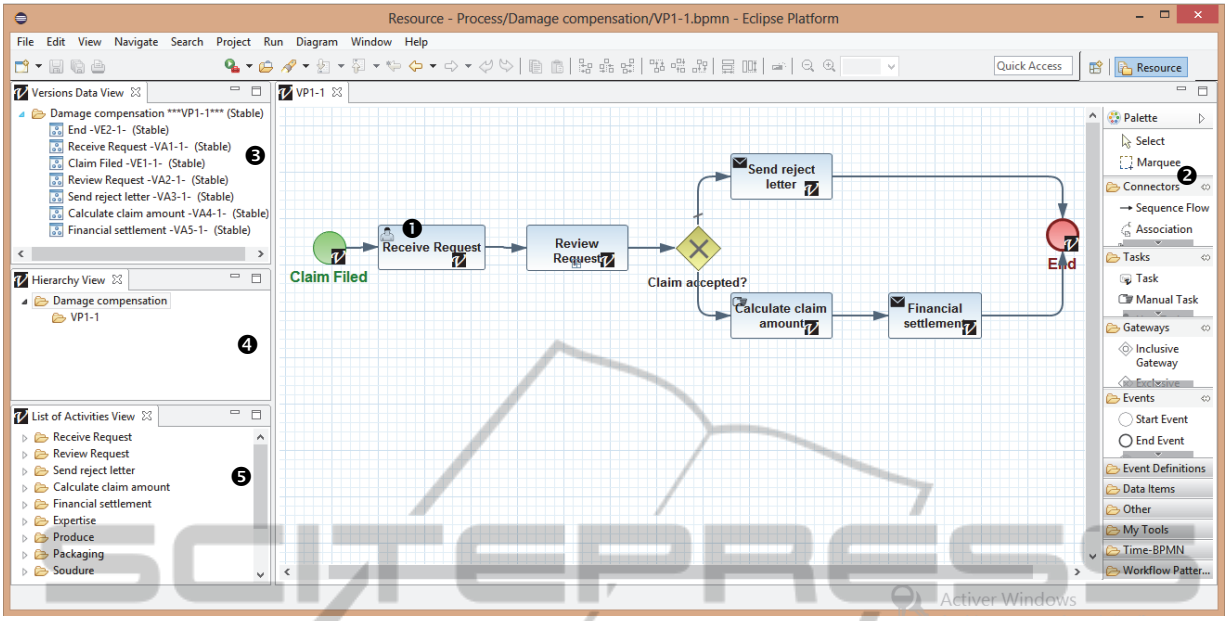

Figure 6: BPMN4V-Modeller overview.

versions of activities, each being decorated with the same icon.

Regarding the added Eclipse views, they mainly display versions details. Actually, the *Versions Data view* gives the corresponding: process name, its id and its state; it also details the activities and eventsthat make it up. More precisely, the Versions Data view presented in Figure 6 part❸ indicates that the active tab contains the version of process identified by VP1-1, which is a stable version, and which corresponds to the first version of the Damage Compensation process.

Moreover, this view indicates (i) that this version of process is composed of the first versions of *Claim Filed (VE1-1)* and *End (VE2-1)*events and of the first versions of the following activities: *Receive Request (VA1-1)*, *Review Request (VA2-1)*, *Send reject letter(VA3-1)*, *Calculate claim amount (VA4-1)*, and *Financial settlement (VA5-1)* and (ii) that all these versions are in the stable state.

The *Hierarchy* view aims to show the derivation hierarchy of the selected versionable concept of the active drawing canvas. In Figure 6 part❹, the *Hierarchy* view shows the derivation hierarchy of the Damage Compensation process since the selected versionable concept is this process.

Finally, the *List of Activities* view, presented in Figure 6 part❺, shows all the modelled activities and their corresponding versions.

4.3 Implementing Dynamic Aspects of Version Management

In order to implement the dynamic aspects of versions in BPMN4V-Modeller, we propose contextual menus and Eclipse dialogs. Actually, we added the *Handle Version* contextual menu on each versionable concept. This menu implements the operations presented before (*cf.* section 3.2). Therefore, we provide Eclipse dialogs to update and to validate working versions and to derive stable ones. Regarding the Create operation, it can be performed using either the tool palette to create new activities, events, etc., or the *Create Process* dialog to create new processes.

In order to illustrate the implementation of these operations more clearly, we detail in the following how to create, validate, derive, and update versions of processes using the BPMN4V-Modeller, considering the two versions of the Damage Compensation process previously presented.

To model the first version of this process, the designer has to create a new process using the *Create Process* dialog. Then, he has to define the versions of activities and events that make up the process, according to two possible scenarios: either he can create a new activity (or event) from the tool palette, or he can reuse an existing version of activity (or event) from the *List of Activities* view. Figure 7 gives a UML sequence diagram which details the interaction between the user and

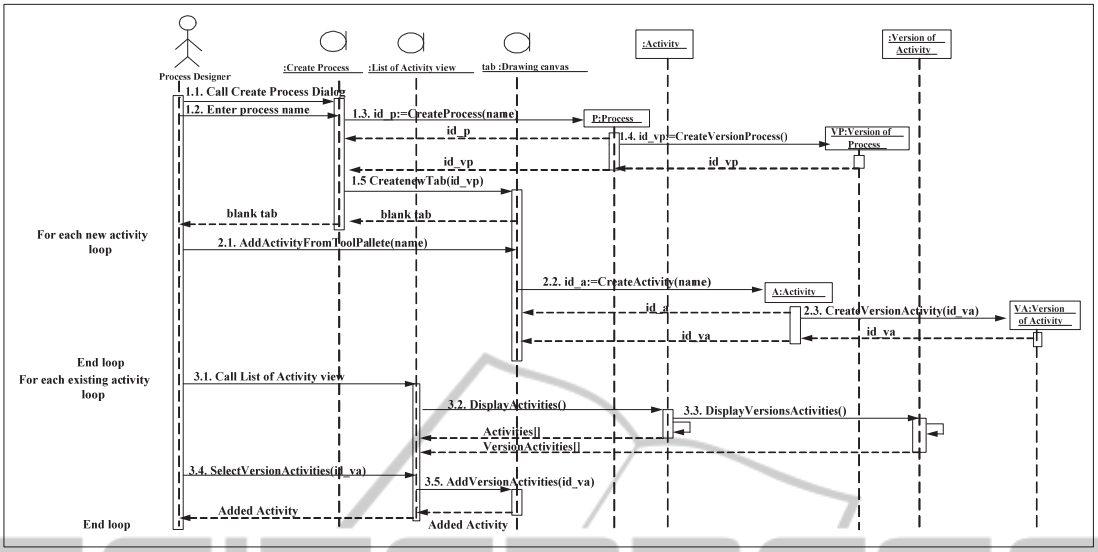

Figure 7: Sequence diagram for creating a new process version.

BPMN4V-Modeller illustrating (i) the creation of a new process and thus its first version, and (ii) the definition of two versions of activities for this process version.

Once the first version is defined, the designer can use the *Validate* command from the *Handle Version* contextual menu to validate this version. Thus the version becomes stable and consequently changes are no longer possible on this version.

Figure 8 presents the result of (i) the creation of the first version of the damage compensation process, (ii) the definition of its components (including versions of events and activities), and (iii) its validation (the version state moves from working to stable).

In order to design the second version of the damage compensation process (*cf.* Figure 5(b)), the designer has to first derive the first version of this process (*i.e.*, *VP1-1*), and then to perform the necessary changes. To do so, the designer can use either the *Derive Process Version* dialog or the *Derive* command from the *Handle Version* contextual menu. In each case, the derivation results in the creation of a new tab that contains a copy of the derived version.

More precisely, when the designer derives *VP1-1,* a new tab named *VP1-2* appears in the drawing canvas. *VP1-2* is a working version that corresponds to the second version of the damage compensation process and it has the same definition as *VP1-1.*

Figure 8 part ❶ is a screenshot illustrating the *Derive Process Version* dialog used to derive *VP1-1*, and figure 8 part ❷ shows the result of the derivation of this version.

Moreover, Figure 9 presents a UML sequence Diagram detailing the interaction between a process designer and BPMN4V-Modeller for deriving a version of process using *Derive Process Version* dialog.

However, as indicated before, the new created process version VP2-1 has to be updated to be consistent with the process definition presented in Figure5 (b). Updating a process version can be performed by (i) adding versions of activities (or events), (ii) deleting existing versions of activities.
(or events), (iii) modifying existing versions of activities (or events), or (iv) modifying coordination of the process by adding or deleting gateways and sequence flows. More precisely, the addition of versions of activities can be performed using the Define Process Version of Component dialog. This dialog displays details about versions of activities (or events) previously modelled (name, derivation hierarchy…) and permits to add the selected version to the active drawing canvas. Figure 10 is a UML sequence diagram detailing interaction between a process designer and BPMN4V-Modeller when updating a version of process using the Define Process Version of Component dialog.

The modification of existing versions of activities can be performed using the *Update Activity Version* dialog. This dialog displays details about the selected activity version and permits specific modifications available to activity versions: modifying type of activity, adding or deleting version of information, adding or deleting version of resources.

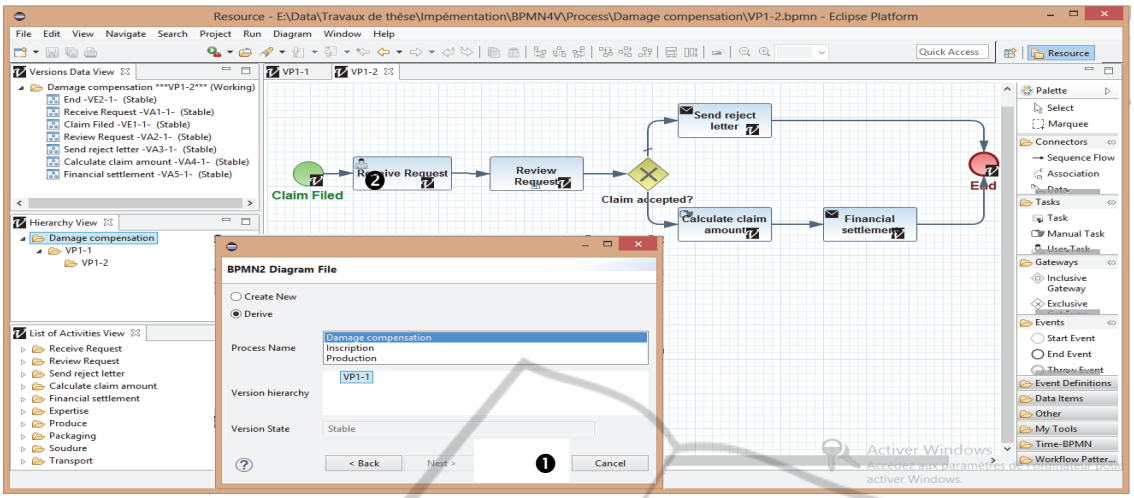

Figure 8: Derivation of VP1-1.

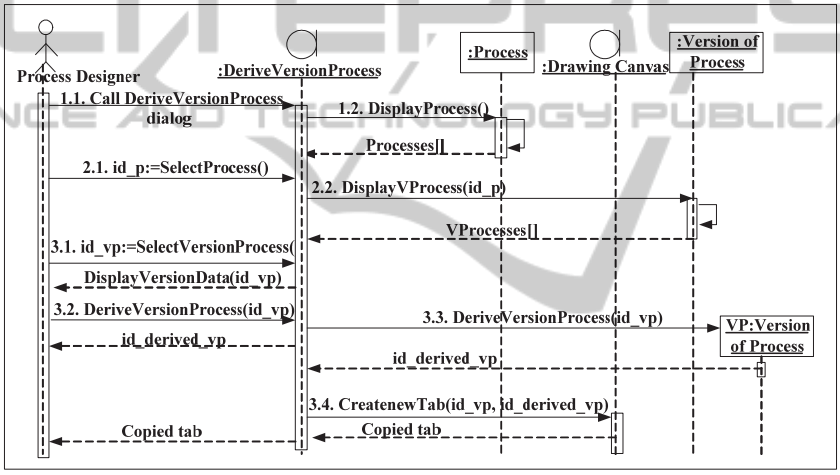

Figure 9: Sequence Diagram for Process Version Derivation using the Derive Process Version dialog.

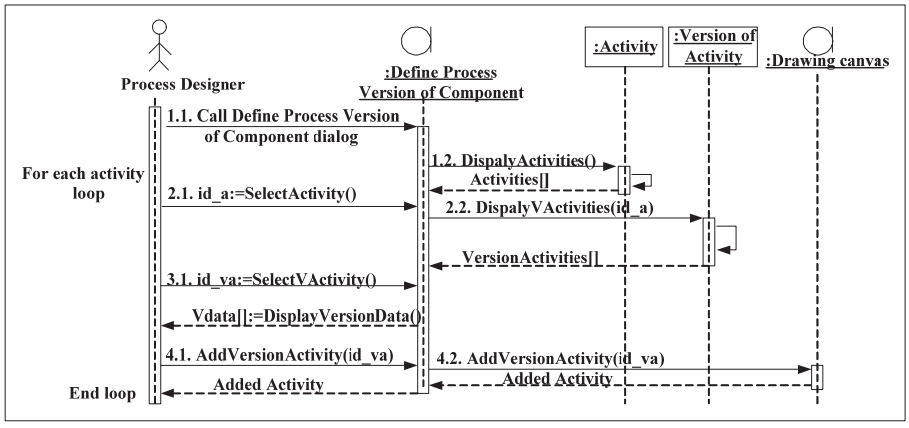

Figure 10: Sequence Diagram for Process Version Update.

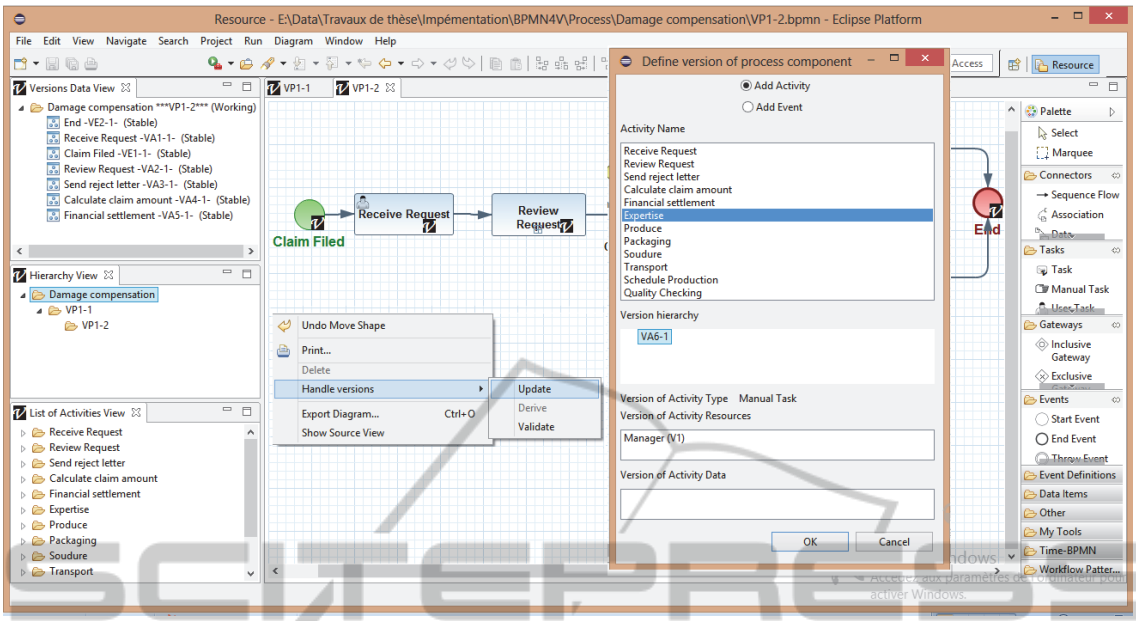

Figure 11: Update VP1-2 using Define Process Version of Component Dialog.

Figure11 gives more details about updating *VP2-1*. It presents how we add the *Expertise* activity to *VP2-1*. More precisely, the *Update* command of the contextual menu, shown in this figure, performs *Define Process Version Component* dialog used to add the first version of this activity.

Due to lack of space, we do not give additional details about process version update.

5 CONCLUSION

This paper has presented BPMN4V, an extension of BPMN addressing process adaptation issue using versions.

More precisely, this paper has illustrated how to model versions of processes in BPMN 2.0, in order to describe process variability, taking into account the behavioral, informational and organizational dimensions of processes. To do so, the BPMN 2.0 meta-model has been extended to integrate version modelling capability and therefore to be able to define variability of activity, event, sub process, process, resource, resource role and data. Moreover, the paper has also dealt with dynamic aspects of process version management, defining state charts for process versions and the corresponding operations. Finally, the paper has introduced BPMN4V-Modeller, a tool supporting the modelling of BPMN business process versions, implementing the dynamic aspects of version management, and defining a graphical notation for process versions. An example is used to illustrate the creation, derivation, validation and update of two process versions (including versions of activities and versions of events).

The advantages of our contribution are the following. First, and as defended in the introduction and in (Zhao and Liu, 2007) (Chaâbane et al., 2009); (Ben Said et al., 2014a), adding such version modelling capability to BPMN makes the modelling of process variability easier. Secondly, the proposed solution promotes the reuse of versions. Thus the designer is not required to provide the entire definition of the modelled processes. He can reuse versions (of sub process, of activities, of events, etc.) previously modelled. Finally, our contribution extends the main process version contributions found in literature as follows:

- It takes into account not only the behavioural dimension of processes but also their informational and organizational dimensions when dealing with process variability: indeed, these dimensions have to be taken into account to have a comprehensive view of process adaptation,
- It defines the dynamics aspects of version management, mainly identifying operations for process version management and indicating when these operations can be performed,
- It implements these propositions in BPMN4V-Modeller, a specific tool intended to business process designers.

Our future works will take three directions. First, we will evaluate BPMN4V-Modeller in a specific evaluation workshop involving BPMN practitioners and master degree students (having knowledge in business process management). Second, we will address inter-organizational process variability, adding versions to BPMN 2.0 collaborations. Third, we will integrate the notion of context in our proposition in order to model another interesting dimension of versions of processes, the *why* dimension, which supports the explicit definition of situations in which process versions have to be used (Ben Said et al., 2014b), (Nurcan and Edme, 2005).

REFERENCES

Kradolfer, M., Geppert, A., 1999. Dynamic workflow schema evolution based on workflow type versioning and workflow migration. *International Conference on Cooperative Information Systems*, Edinburgh, Scotland, September 1999, pp. 104–114.

Zhao, X., Liu, C., 2007. Version Management in the Business Change Context. *International Conference on Business Process Management*, Brisbane, Australia, September 2007, pp. 198–213.

Chaabane, M. A., et al., 2009. Versions to Address Business Process Flexibility Issue. *European Conference on Advances in Databases and Information Systems*, Riga, Latvia, September 2009, pp. 2–14.

Smith, H., Fingar, P., 2003. Business Process Management: the Third Wave. *Megan-Kiffer Press*, 2003

Dumas, M., et al., 2005. Process-Aware Information Systems: Bridging People and Software through Process Technology. *Wiley & Sons*, 2005.

Weske, M., 2007. Business Process Management: Concepts, Languages, *Architectures*. Springer-Verlag 2007.

Casati, F., et al., 1996. Workflow Evolution. *International Conference on the Entity Relationship Approach*, Cottbus, Germany, October 1996, pp. 438–455.

Nurcan, S., 2008. A Survey on the flexibility Requirements related to Business Process and Modelling Artifacts. International Conference on System Sciences, Waikoloa, Big Island, Hawaii, USA, January 2008, pp. 378–387.

Schonenberg, H., et al., 2008. Process Flexibility: a Survey of Contemporary Approaches. *CAiSE Workshop*, Montpellier, France, June 2008, pp. 16–30.

Weber, B., et al., 2009. Dynamic Process Lifecycle Support: a Survey on Dynamic Changes in Process-Aware Information Systems. *Computer Science, Research and Development*, Vol. 23, n°2, 2009, pp. 47–65.

Adams, M., et al., 2006. Worklets: a Service-Oriented Implementation of Dynamic Flexibility in Workflows. *International Conference on Cooperative Information Systems*, Montpellier, France, October 2006, pp. 291–308.

Lu, S., et al., 2009. Defining Adaptation Constraints for Business Process Variant. *International Conference on Business Information Systems*, Poznan, Poland, April 2009, pp. 145–156.

Hallerbach, A., et al., 2010. Capturing Variability in Business Process Models: the Provop Approach. *Software Maintenance*, Vol. 22, n°6-7, June 2010, pp. 519–546.

OMG, 2011. Business Process Model and Notation (BPMN) Version 2.0. OMG Document Number: formal/2011-01-03, available at: http://www.omg.org/spec/ BPMN/2.0, 2011.

Korherr, B., List, B. 2007. Extending the EPC and the BPMN with Business Process Goals and Performance Measures. *International Conference on Enterprise Information Systems*, Funchal, Madeira, Portugal, June 2007.

Santos, E., et al., 2010. Configuring the Variability of Business Process Models Using Non-Functional Requirements. CAiSE Conference on Business Process Modeling, *Development and Support*, Hammamet, Tunisia, June 2010, pp 274–286.

Ben Said, I., et al., 2010. A Model Driven Engineering Approach for Modelling Versions of Business Processes using BPMN. *International Conference on Business Information Systems*, Berlin, Germany, May 2010, pp. 254–267.

Ben Said, I., et al., 2014a. Extending BPMN2.0 Meta-models for Process version Modelling. *International Conference on Enterprise Information Systems*, Lisbon, Portugal, May 2014, pp. 384–393.

Ben Said, I., et al., 2014b. Context-Aware Adaptive Process Information Systems: The Context-BPMN4V Meta-Model. *International Conference on Advances in Databases and Information Systems*, Ohrid, Macedonia, September 2014, pp. 366–382.

Nurcan, S., Edme, M. H. 2005. Intention Driven Modelling for Flexible Workflow Applications. *Software Process: Improvement and Practice*, Vol. 10, n°4, 2005, pp. 363–377.