Extending BPMN 2.0 Meta-models for Process Version Modelling

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

Keywords: Business Processes, Versions, BPMN 2.0.

Abstract: This paper introduces BPMN4V (BPMN for Versions), an extension of BPMN for modelling variability (flexibility) of processes before their use in an organizational context or before their publication over the cloud as services. More precisely, this paper motivates the importance of modelling variability of processes using versions and introduces the versioning pattern to be used to reach this objective. It also presents BPMN4V, giving provided extensions to BPMN2.0 meta model, both considering versions of intra and inter-organizational processes. An example illustrating the instantiation of the proposed meta-model is given for each kind of process.

1 INTRODUCTION

With the advent of cloud computing and the emergence of BPaaS (Business Process as a Service) (Wang et al., 2010), enterprises and organizations outsource their processes onto the cloud. Thus, the cloud is now viewed as an environment in which several processes may be used by several enterprises which share common infrastructures and services. As a consequence the cloud is the support for process discovery, which requires process analysis abilities in order to decide to use a given process or not (Aalst, 2013a). Of course, all the dimensions (organizational, informational and behavioural) of a process have to be considered for this decision-making.

Among the different processes spread over the cloud, some are close to each other: they can be viewed as versions or variants of a same process. Instead of discovering this variability by analyzing the different dimensions of the considered processes or by analysing the log files resulting from their execution using mining techniques (Aalst, 2013b), we propose in this paper to define it since the process modelling step, *i.e.,* before their outsourcing onto the cloud. Indeed, discovering the variability is a difficult task and the solutions of the literature are very few and not really convincing -e.g. (Luengo and Sepulveda, 2011).

Process variability, and more generally process-adaptation issue, has been deeply investigated these last years. Several typologies for classifying variability capabilities of languages or notations have also been introduced (Schonenberg et al., 2008), (Nurcan, 2008), (Weber et al., 2008), (Andonoff et al., 2013), and three main types of variability are identified: *variability by design* (or flexibility) for handling foreseen changes in processes, *variability by deviation* for handling occasional unforeseen changes and where the differences with initial process are minimal, and finally, *variability by evolution* for handling unforeseen changes in processes, which require occasional or permanent modifications in their schemas. However, existing languages and notations introduced for modelling this variability (Kradofler and Grepper, 1999), (Lu and Shadiq, 2006), (Zhao and Liu, 2007), (Hallerbach et al., 2008), (Lu et al., 2009), (Hallerbach et al., 2010), (Chaâbane et al., 2011), (Angles et al., 2013) are incomplete as either they deal with only one type of variability and do not address these three types of variability in a coherent framework, or they mainly focus on the behavioural dimension of processes, or they are too specific with low degree chance to be used, or finally they are dedicated to specific domains.

On the other hand, in a previous paper, we have defended the importance of versions to deal with this variability issue and more precisely to deal with the different types of variability (Chaâbane et al., 2009). As a consequence, and in order to define a notation that will perhaps be used in the future, we propose to

Ben said I., Chaâbane M., Andonoff E. and Bouaziz R..
Extending BPMN 2.0 Meta-models for Process Version Modelling.
DOI: 10.5220/0004891103840393
In *Proceedings of the 16th International Conference on Enterprise Information Systems* (ICEIS-2014), pages 384-393
ISBN: 978-989-758-029-1

extend BPMN2.0 (OMG, 2011), which is considered as the de-facto standard for modelling processes, in order to make the modelling of this variability possible. More precisely, we extend the BPMN2.0 meta model to support the modelling of process versions, considering all the dimensions of processes (*i.e.,* organizational, informational and behavioural dimensions) in the context of intra and inter-organizational processes.

We defend the use of BPMN2.0, instead of EPC (Scheer et al., 2005) or UML activity diagrams (Engles et al., 2005) for instance, because it is a promoted standard and it can be easily understood by all business users: business analysts that design processes, the technical developers that are responsible of technical implementation of those processes, and finally people (actors or process owners) that manage and control those processes (Weske, 2007).

As stated before, the paper contribution is an extension of BPMN2.0 to support the modelling of inter or intra-organizational process variability trough versions. This extension is called BPMN4V (BPMN for Versions).

The paper is organized as follows. Section 2 gives a brief state of the art about modelling variability of processes. It then presents our approach for modelling this variability and introduces the versioning pattern we use for that. Sections 3 and 4 are respectively dedicated to the presentation of BPMN2.0 meta-model extensions for modelling versions of intra-organizational processes (private processes) and versions of inter-organizational processes. In both sections, an example illustrates the definition of process versions. Finally, section 5 concludes the paper, highlighting its contribution and giving directions for future works.

2 MODELING PROCESS VARIABILITY USING VERSIONS

This section first introduces related works and then defends and presents our versioning approach for modelling process variability.

2.1 Related Works

Variability has been the focus of numerous works in the BPM area during the last decade. As indicated before, several typologies for classifying variability capabilities of notations or systems have been introduced these last years (Nurcan, 2008), (Schonenberg et al., 2008), (Weber et al., 2008) (Andonoff et al., 2013). These works distinguish three types of process variability: *variability by design* for handling foreseen changes in processes, *variability by deviation* for handling occasional unforeseen changes in processes, where the differences with the initial process are minimal, and finally, *variability by evolution* for handling unforeseen changes in processes, which require significant occasional or permanent changes. In addition to the three types of variability, these works also considers two main times for changing processes:

- *design-time*, which corresponds to expected changes that can be foreseen; in this case variability can be a priori defined as it is possible to model the whole process at design-time;
- *run-time*, which corresponds to unexpected changes that can not be foreseen; in this case, variability cannot be a priori defined as the process will be changed at run-time.

Several contributions have been proposed to address the variability issue. For instance, the notion of variant is introduced in (Hallerbach et al., 2008) and (Hallerbach et al., 2010). A variant is an adjustment of a basic process model to unexpected requirements of process contexts appearing at run-time. As a consequence, these work mainly deal with variability by evolution, at run-time. On the other hand, several version-based meta-models for capturing process changes have been proposed in the literature (Kradofler and Geppert, 1999), (Zhao and Liu, 2007), (Chaâbane et al., 2010), (Chaâbane et al., 2011). For instance (Zhao and Liu, 2007) proposed to model versions of processes as direct graphs named VPG (Version Preserving directed Graph). The nodes of VPG are activities of the process while the arcs of VPG define coordination between these activities. (Zhao and Liu, 2007) also proposed operations to modify the VPG: create node, delete node, replace node, create arc, etc. However, this work mainly focuses on the behavioural dimension of processes, neglecting the informational and organizational dimensions which are however important to have a comprehensive view of processes (Aalst et al., 2003). (Chaâbane et al., 2010) compensated this weakness but the proposed notation is too specific and has no chance to be used.

Another interesting work is the one of (Weber et al., 2008). The authors introduced the notion of change pattern to define process changes. They

distinguish patterns to deal with variability at design-time and run-time. However, again the notation is too specific, too technical, and thus has low chance of being used.

2.2 Our Approach for Process Variability Modelling

We consider that the notion of version subsumes the notion of variant. As illustrated in Figure1 below, a version corresponds to one of the significant states a process may have during its life cycle. So, it is possible to describe the variability of a process through its different versions. These versions are linked by a derivation link; they form a version derivation hierarchy. When created, a process is described by only one version. The definition of every new version is done by derivation from a previous one: such versions are called derived versions. Of course, several versions may be derived from the same previous one: they are called alternatives or variants. The derivation hierarchy looks like a tree if only one version is directly created from a process entity, and it looks like a forest if several versions are directly created from the considered process entity.

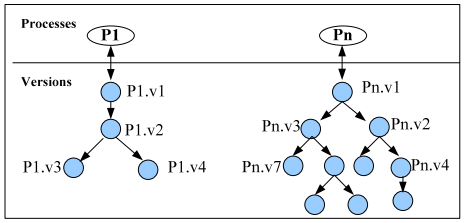

Figure 1: Versions to Model Process Variability.

To sum up, when considering versions, we model both process evolution and process alternative (*i.e., variant*) to describe process variability.

2.3 Versioning Pattern

This paper defends the idea of using versions to make BPMN2.0 notation multi-versionable. Thus we have defined a versioning pattern to support versions for the main concepts of BPMN.

The versioning pattern we propose is very simple: it includes only two classes: "Versionable" class and "Version of Versionable" class, and two relationships: "Is_version_of" and "Derived_From" as illustrated in Figure 2. A versionable class is a class for which we would like to handle versions. In addition we define a new class which contains versions, called "*Version of Versionable*".

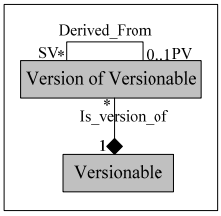

Figure 2: Versioning Pattern.

The "*Is_version_of*" relationship links a class to its corresponding versions. The "*Derived_From*" relationship allows for building version derivation hierarchies (cf. Figure 1). This latter relationship is reflexive and the semantic of both relationship sides is the following: (i) a version (SV) succeeds another one in the derivation hierarchy, and (ii) a version (PV) precedes another one in the derivation hierarchy. Regarding properties of a "*Version of Versionable*" class, we introduce the classical version properties, *i.e.,* version number, creator name, creation date and status.

3 MODELING VERSIONS OF INTRA-ORGANIZATIONAL PROCESSES

BPMN2.0 allows the creation of three basic types of sub-models within an end-to-end process (OMG, 2011):

- *Process* (*private* and *public*): a *private process* is internal to a specific organization with the objective of carrying out work. As for a *public process,* it represents the interactions between a private process and another process or business entity.
- *Collaboration*: a *collaboration* depicts the interactions between two or more *private processes*.
- *Choreography*: a *choreography* is an extended type of collaboration that defines the sequence of interaction between business entities.

Intra-organizational processes can be modelled using BPMN2.0 through private process concepts. A private process describes a sequence of activities performed within an organization in order to carry out an objective. It is depicted as a directed graph.

This section first introduces BPMN concepts for modelling private processes and then illustrates how we use the previous versioning pattern to make some of these concepts versionable.

3.1 Concepts to Model Private Processes

Figure 3 below introduces BPMN 2.0 meta-model concepts useful to model private processes (OMG, 2011).

Regarding the behavioural dimension of processes, *FlowElementContainer* is an abstract class used to define the superset of elements in BPMN diagrams. An object *FlowElementContainer* contains *SequenceFlows*, *FlowNodes* (*Gateways*, *Events*, and *Activities*), and *Data Objects*. A *SequenceFlow* is used to show the order of *FlowNodes* in a process. A *Gateway* is used to control how *SequenceFlows* interact within a process. An *Event* is something that "happens" during the course of a process. It affects the flow of the process and usually has a cause or an impact, and in general requires or allows a reaction. More precisely, an *Event* can catch a trigger "*CatchEvent*" which means it reacts to something, or it can throw a result "*ThrowEvent*". An *Event* can be composed of one or more *EventDefinitions*. An *EventDefinition* refers to the triggers of *CatchEvents* and to the Results of *ThrowEvents*. There are many types of *Event Definitions*: *ConditionalEventDefinition, TimerEventDefinition,* etc. 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 *Sub Process* is refined via *Activities*, *Gateways*, *Events*, and *SequenceFlows*. A *Task* is used when the work is elementary (*i.e.,* it cannot be more refined). BPMN2.0 identifies different types of *Tasks* such as *Service Task*, *User Task*, *Manual Task*, *Send Task* and *Receive Task*.

Regarding the organizational dimension of processes, an activity is accomplished by *ResourceRoles*. A *ResourceRole* can refer to a *Resource*. A *Resource* can define a set of parameter called *ResourceParameters*. A *ResourceRole* can be a *Performer*, which can be a *HumanPerformer*, which can be in turn a *PotentialOwner*. The *Performer* class defines the resource that will perform or will be responsible for an *Activity*. A *Performer* can be human (*HumanPerformer*), *i.e.,* it defines people that are assigned to *Activities*. A *PotentialOwner* is a specialization of *HumanPerformer*, used to define persons who can claim and work on *User Task*.

Regarding the informational dimension of processes, *ItemAwareElement* references elements 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*.

DataObject, *Property* and *DataObjectReference* are ways for modelling data of each process. *DataObject* elements have a graphic representation on a process diagram, while *Properties* have not. *DataObjectReference* allows the reuse of a *DataObject* in the same diagram. *DataStore* provides a mechanism for *Activities* to retrieve or update stored information used in a process. *DataInput* and *DataOutput* are used to model data needed and produced during an execution or as a result of execution. Data requirements are captured as *DataInputs* and *InputSets*. Data that are produced are captured using *DataOutputs* and *OutputSets*. These elements are aggregated in the *InputOutputSpecification* class.

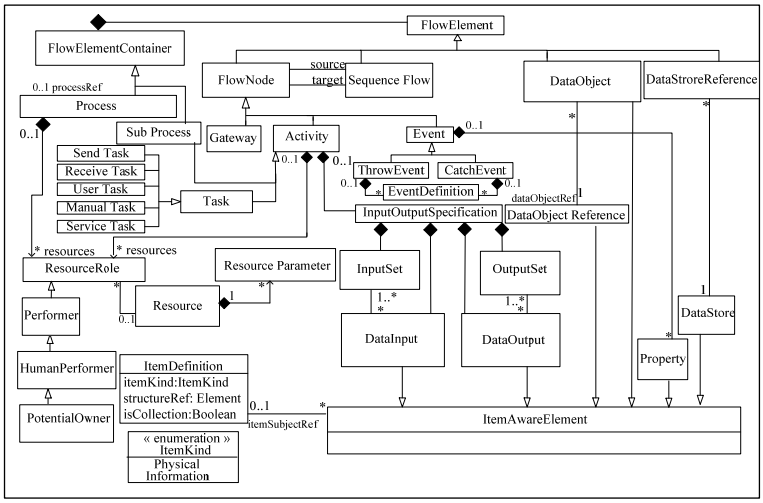

Figure 3: Extract of BPMN2.0 Meta-Model for Modelling Private Processes.

3.2 Extending BPMN Meta Model for Modelling Versions of Private Processes

This section explains the proposed BPMN extensions to consider versions of processes. The idea is to use the versioning pattern introduced before (cf. Figure 2) to make some classes of the BPMN meta-model versionable. Figure 4 below presents the resulting meta model.

We propose to handle versions for six classes: *Process*, *Sub Process*, *Event*, *Activity*, *ItemAwareElement* and *Resource*, in order to take into account process variability. This variability is expressed by the different versions of these class instances, each one representing a significant state (alternative, *i.e.,* variant, or derived) of the considered element (*e.g.,* a process). A new version of an element (*e.g.,* a process or a resource) is defined according to the changes occurring to it: these changes may correspond to the adding of a new information (property or relationship) or to the modification or the deletion of an existing one. Actually, these changes can affect all the dimensions of a process: (i) its behavioural dimension to redefine how the process is achieved (process, sub process, activity and event), (ii) its organizational dimension to redefine the roles invoked by the process (*Resource*) or, (iii) its informational dimension to redefine the used or produced information (*ItemAwareElement*). The general idea is to keep track of changes occurring to elements participating to the description of the way business is carried out.

Regarding the organizational dimension of a process, we create a new version of *Resource* when we change its parameters. For instance, a *Manager* resource may be defined using two parameters: name and experience. A new version of *Manager* may be defined if it becomes necessary to consider another parameter (e.g. region of the manager) and this definition can lead to the definition of a new process in which this resource is involved. We also create versions of *Resource* when there is a change in its privileges. For instance, an *Employee* is a *HumanPerformer* resource that performs three activities. Because some activities of the process in which this employee is involved become automatic, the employee can perform anymore only two activities. A new version of the employee has then to be defined.

Regarding the informational dimension of processes, and more particularly *ItemAwareElement*, we consider that changes in the structure and/or the type of an *ItemDefinition* results in the creation of a new version. For example, if *Report* is an *ItemAwareElement* corresponding to a paper document (*Itemkind* is a *Physical* data), and if after technical changes it becomes an electronic document (*Itemkind* becomes an *Information* data), then a new version of *Report* has to be created.

Regarding the behavioural dimension of processes, several classes can gather versions: *Event*, *Activity*, *Sub Process* and, of course, *Process*. More precisely regarding activities, we create a new

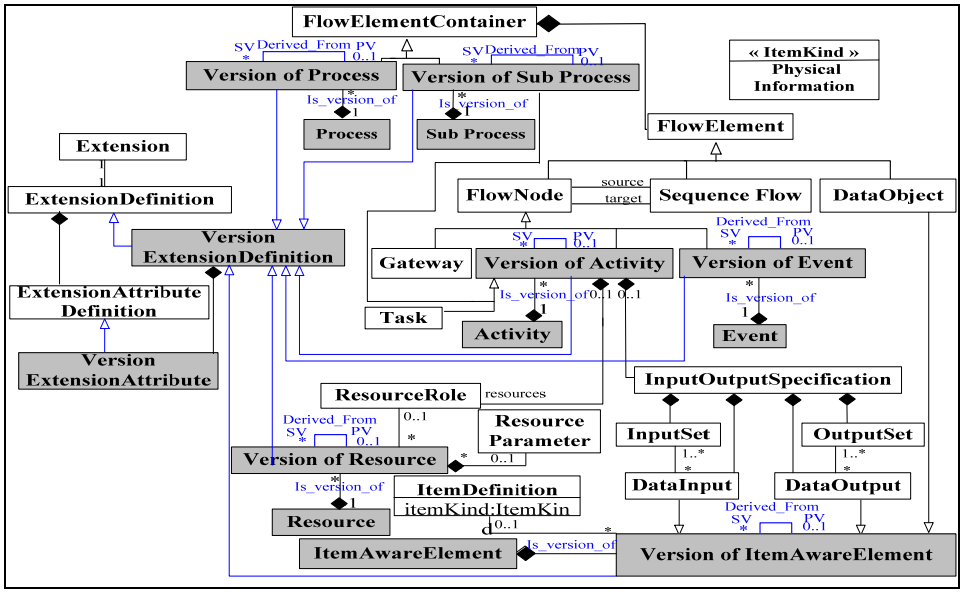

Figure 4: Extending BPMN2.0 Meta-model for Modelling Versions of Private Processes.

version of an activity when there are changes in the type of the activity (a manual activity becomes a service one), in the involved resources, or in the required or produced data. Regarding events, we create a new version of an event when there is change in the associated *EventDefinition*. For instance, if an *Alert* is a signal event (*i.e.,* it has a *SignalEventDefinition),* and if after technical changes it becomes a message event (*i.e.,* it has a *MessageEventDefinition)*, then a new version of *Alert* has to be created. Regarding sub processes and processes, we create new versions when there are changes in the involved activities or in the way they are linked together (used patterns are changed).

On the other hand, BPMN meta-model provides extension mechanisms through classes *Extension*, *ExtensionDefinition* and *ExtensionAttributesDefinition* and each proposed extension should be assigned to these classes (OMG, 2011). In our case, we propose to add the *VersionExtensionDefinition* class which is an abstract super class for all versions of versionable classes. The *VersionExtensionDefinition* class has a set of attributes described into *VersionExtensionAttribute* class. Actually, a version class contains specific attributes such as version number, creator name, creation date and status. Each of *Version of Versionable* class introduced to model process variability is a sub-class of the *VersionExtensionDefinition* class.

3.3 Example

We illustrate in Figure 6 the instantiation of our meta-model according to the damage compensation process of an insurance company. This process is shown in Figure 5. Basically, it contains five activities: Receive Request, Review Request, Send reject Letter, Calculate claim amount and Financial settlement. For simplification reasons, only the behavioural dimension of the process is presented.

The first version of this process is represented in the left part of Figure 5. This version starts when receiving a claim. The checking of the claim can be hold only if *Conditions of Review* is satisfied. *Conditions of Review* is a conditional intermediate event that refers to the condition: *the claim date must not exceed 7 days from the accident date*. 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), and the financial service prepares the financial settlement.

The insurance company has modelled a second version of this process to take into account a new law that imposes the insurance companies to make an expertise when the damage amount exceeds 1000$. The right part of Figure 5 illustrates this second version of the process introducing the *Expertise* activity and both modifying the intermediate *Conditions of Review* event and the *Financial settlement* activity (their type have changed). To sum up, the variability of the damage compensation process is defined by two versions of the process itself, two versions of *Conditions of Review* event and two versions of the Financial settlement activity: the first version of this activity holds for the first version of the process while the second one holds for the second version of the process. In addition, the sequence flows and patterns have been modified in the second version of the process.

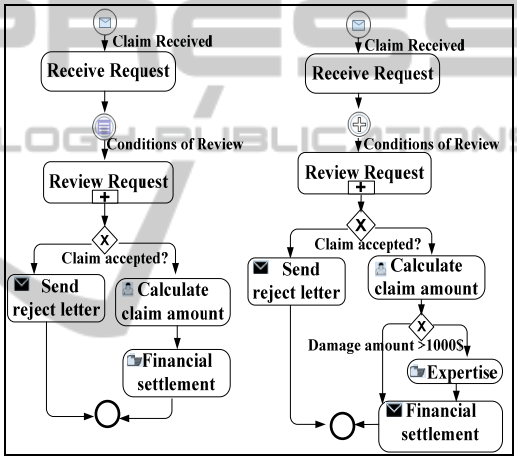

Figure 5: Variability of the Damage Compensation Process.

Regarding variability of the *Conditions of Review* event, in the first version of the process, it is a conditional event referring to the condition: the claim date must not exceed 7 days from the accident date (*i.e.,* it has a *ConditionnalEventDefinition*), while in the second process version, we add another *EventDefinition* the *TimerEventDefinition* which indicates that the checking of the claim have to be hold in the beginning of the week (*i.e.,* in Monday, Tuesday or Wednesday). As a consequence, the *Conditions of Review* event becomes a multiple parallel event that has two *EventDefinitions*: *Conditional EventDefinition* (the claim date must not exceed 7 days from the accident date) and *TimerEventDefinition* (claim checking have to be hold in the beginning of the week).

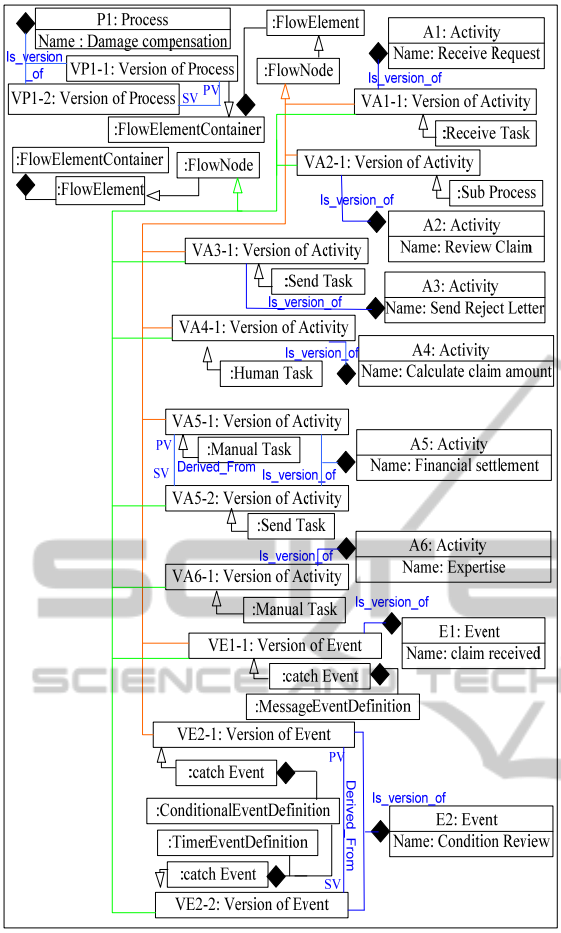

Figure 6: Extract of BPMN4V Meta-Model Instantiation according to the Damage Compensation Process.

4 MODELING VERSIONS OF INTER-ORGANIZATIONAL PROCESSES

An inter-organisational process is a collaboration between organisations (each one represented by a process) in order to carry out a common business target. BPMN2.0 defines collaboration diagrams for modelling processes crossing organisations boundaries. These diagrams include an explicit representation of permanent interactions between the involved entities. These interactions are defined as message flows, *i.e.,* messages exchanged between involved entities (OMG, 2011). Note that BPMN collaboration diagrams are used to model *tight* inter-organisational processes, in which the collaboration infrastructure is well-established and the involved partners are known before process execution (Divitini et al., 2001).

We propose to introduce the notion of version in such diagrams to deal with variability of inter-organizational processes. As a consequence, this section first introduces BPMN concepts for modelling collaboration diagrams and then details how we extend such diagrams to represent versions of inter-organizational processes.

4.1 BPMN Meta-model for Modelling Collaboration Diagrams

Figure 7 below gives the main BPMN concepts for modelling collaboration.

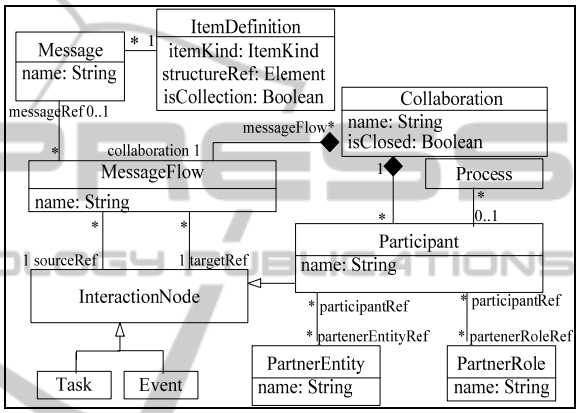

Figure 7: Extract of the BPMN2.0 Collaboration Meta-Model (OMG, 2011).

A *Participant* represents partner entities (e.g., a company) and/or partner roles (e.g., a buyer, seller, or manufacturer) that are involved in a collaboration. A *Participant* is often responsible for the execution of a *Private process*. During a collaboration, participants are prepared to send and receive *Messages*. A *MessageFlow* illustrates the flow of messages between two *InteractionNodes*. An *InteractionNode* element is used to provide a single element as the source or the target of a *MessageFlow*. An *InteractionNode* can be a *Participant*, a *Task* or an *Event*.

4.2 Extending BPMN Meta-model for Modelling Versions of Collaboration Diagrams

We propose to use the versioning pattern introduced before to make some classes of the previous meta-model versionable. Figure 8 below presents the resulting meta-model.

We propose to handle versions for only two classes: *Collaboration* and *Message*, in order to take into account inter-organizational process variability.

A version of collaboration contains a set of participants, each one refering to a version of a process. It also contains a set of messages flow, each one refering to a version of message. For the same reason than the one presented in section 3.2, *Version of Message* and *Version of Collaboration* are sub-classes of the class *VersionExtensionDefinition*.

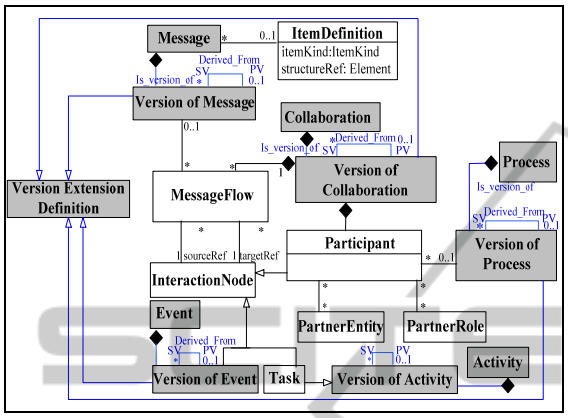

Figure 8: BPMN2.0 Meta-Model for Modelling Versions of Collaboration.

As explained before, a new version of an element (e.g., a collaboration or a message) is defined according to changes occurring to it: these changes may correspond to the addition of information (property or relationship) or to the modification or the deletion of an existing one. A new version of a collaboration may also result from changes of participants. When we add or delete a participant (of a process involved in the collaboration), it is necessary to adapt the current process to this change: we have to integrate the added participant or to overcome the absence of the deleted one.

A new version of a collaboration may also result from message changes. Exchanged messages have an important impact in collaborations flow. Thus, any change in a sent or a received message affects the involved activities, and as a consequence, the involved process. So, when we add (or delete) a message, we have to add (or to delete) a received and a send activity, which leads to change the process schema. Moreover, when we change the structure or the kind of *ItemDefinition* referenced by an existing message, we also have to change the involved sent and received activities.

Finally, a new version of a collaboration may result from a process change. Processes that participate in a collaboration can change their schema. A process can for instance add or delete activities or change the pattern linking these activities. In this case, the other processes involved in the collaboration have in turn to adapt to this change in order to go on ensuring the collaboration.

Figure 9 below summarizes all the situations that lead to change a collaboration diagram.

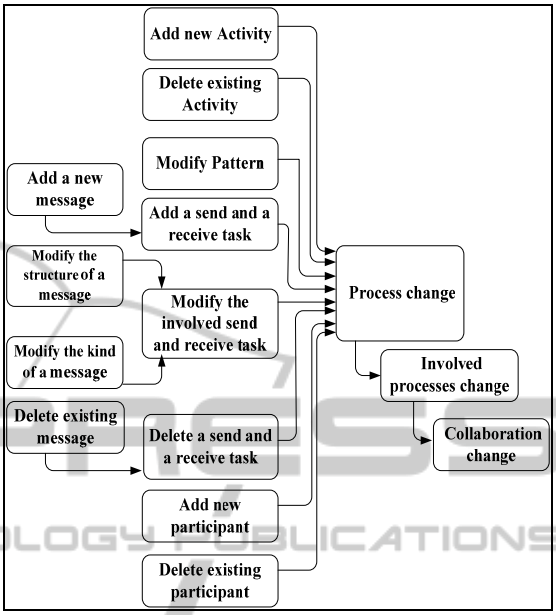

Figure 9: Causes of Collaboration Changes.

4.3 Example

We illustrate in Figure 11 the instantiation of our meta-model using the damage compensation inter-organizational process of the insurance company. Figure 10 gives the third version of the damage compensation process, which extends the example introduced in section 3.3. In this version, the insurance company decides to modify its work strategy subcontracting the expertise activity. Thus, a collaboration between the insurance company and an expert agency is required. The collaboration initiates when a claimant sends a request to the insurance company. After checking, the company sends a reject to the claimant if the request is not accepted. Otherwise, in case where the damage amount exceeds 1000$, an expertise request is sent to an expert. The expert proceeds to the expertise and sends back a report that resumes the expertise results. Based on the received report, the claim amount is calculated and a financial settlement is done.

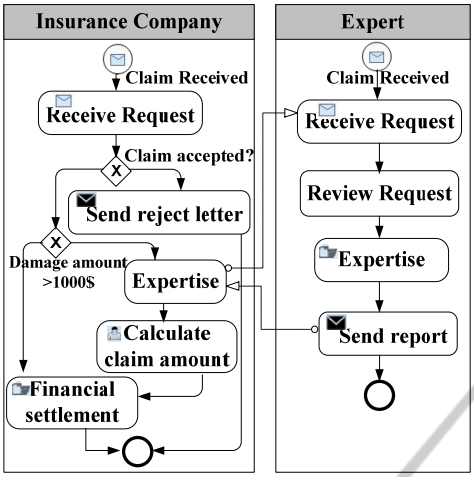

Figure 10: Collaboration between the Damage Compensation Process and the Expert Process.

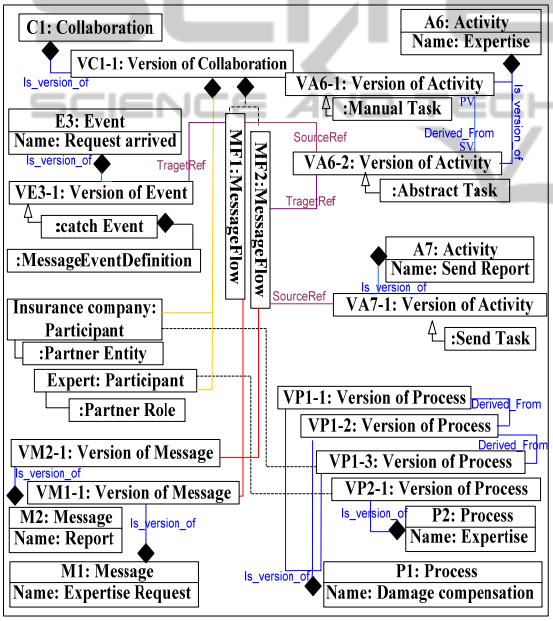

Figure 11: Extract of BPMN4V Meta-Model instantiation according to the Collaboration Example.

5 CONCLUSIONS

This paper has proposed a solution to model variability of intra and inter organizational processes using versions. More precisely, it has proposed BPMN4V, an extension of BPMN2.0 to consider versions of private processes (intra-organizational processes) and versions of collaborations (inter-organizational processes). For each of these kinds of processes, this paper has extended BPMN2.0 meta-models and argued why and when creating versions of elements.

Our future works will take two directions. On the one hand, we will implement the BPMN4V in order to obtain a specific tool for modelling versions of private and collaboration processes. The consequence will be the introduction of a specific graphical notation for versions. On the other hand, we will investigate context of intra and inter organizational processes. Our objective is to give information about (i) version of process goals (Korherr and List, 2006) and (ii) version of process use, *i.e.,* in which circumstances (which situations) do we use a specific version of process instead of another one (Saidani and Nurcan, 2009), (Chaâbane et al., 2010). To sum up, we intend to extend BPMN2.0 in order to incorporate the context dimension.

REFERENCES

Aalst, W. M. P. et al., 2003. Advanced topics in workflow management: issues, requirements and solutions. *Journal of Integrated Design and Process Science*, 7(3), pp. 47-77.

Aalst, W. M. P., 2004. Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management. *Lectures on Concurrency and Petri Nets*, Advances in Petri-Net, Eichstätt, Germany, pp. 1–65.

Aalst, W. M. P., 2013a. Business Process Management: A Comprehensive Survey. Software Engineering (to appear).

Aalst, W. M. P., 2013b. Challenges in Service Mining: Record, Check, Discover. International Conference on Web Engineering, Aalborg, Denmark, July 2013, pp. 1–4.

Andonoff, E., et al, 2013. Adaptation des processus d'entreprise. Chapter 3, in L'adaptation dans tous ses états, Lopisteguy, P., Rieu, D. Roose, P. Edition, Cepadues Editor.

Angles, R., et al., 2013. V-BPMI: a Variability-oriented Framework for Web-based Business Process Modelling and Implementation. *IEEE International Conference on Research and Challenges on Information Science*, Paris, June 2013, pp. 313–324.

Chaâbane, M. A., et al., 2009. Versions to Address Business Process Flexibility Issue. *International Conference on Advances on Database and Information systems*, Riga, Latvia, september 2009, pp. 2–14.

Chaâbane, M. A., et al., 2010. Modélisation multidimensionnelle des versions de processus. *Journal on Ingénierie des Systèmes d'Information*, 15(5), December 2010, pp. 89–114.

Chaâbane, M. A., et al., 2011. Towards a Business Process Versions Query and Manipulation Language. *International Conference on Information Technology and E-Services*, Sousse, Tunisia, August 2011, pp. 12–19.

Divitini M., Hanachi C., Sibertin-Blanc C., «Inter-Organizational Workflow for Enterprise Coordination. Coordination of Internet Agents», chapter 15, Tolksdorf Editors. *Springer Verlag*, 2001.

Engles, G., et al., 2005. Process Modelling using UML. Process-aware Information Systems: Bridging People and Software through Process Technology, Chapter 5, *Wiley and Sons Editor*, pp. 85–117.

Hallerbach, A. et al., 2008. Managing Process Variants in the Process Life Cycle. *International Conference on Enterprise Information Systems*, Barcelona, Spain, pp. 154–161.

Hallerbach, A. et al., 2010. Capturing Variability in Business Process Models: the Provop Approach. *Journal of Software Maintenance 22(6-7)*, pp. 519–546.

Korherr, B., List, B., 2006. Extending the UML 2 Activity Diagram with Business Process Goals and Performance Measures and the Mapping to BPEL, *Proceedings of the 2nd International Workshop on Best Practices of UML*, International Conference on Entity Relationship, Tucson, Arizona, USA, Springer Verlag, November 2006, pp. 7–18.

Kradofler, M. and Geppert, A., 1999. Dynamic Workflow Schema Evolution based on Workflow Type Versioning and Workflow Migration. *International Conference on Cooperative Information Systems*. Edinburgh, Scotland, pp. 104–114.

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

Lu, R., and Sadiq, S., 2006. Managing Process Variants as an Information Resource. *International Conference on Business Process Management*, Vienna, Austria, September 2006, pp. 426–431.

Luengo, D. and Sepulveda, M., 2011. Applying Clustering in Process Mining to find different versions of a business process that changes over time. *Business Process Intelligence Workshop at International Conference on Business Process Management*, Clermont-Ferrand, France, September 2011, pp. 153–158.

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

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

Saidani, O. and Nurcan, S., 2009. Context-Awareness for Adequate Business Process Modelling. *IEEE International Conference on Research Challenges in Information Science*, Fés, Morocco, pp. 177–186.

Scheer, A.W., et al., 2005. Process Modelling using Event-Driven Process Chain. Process-aware Information Systems: Bridging People and Software through Process Technology, Chapter 6, Wiley and Sons Editor, pp. 119–145.

Schonenberg, H., et al., 2008. Process Flexibility: A Survey of Contemporary Approaches. *In the International Workshop on CIAO/EOMAS at International Conference on Advanced Information Systems*. Montpellier, France, pp. 16–30.

Wang M., et al., 2010. Process as a service. *International Conference on Services Computing, Miami*, Florida, July 2010, pp. 578–585.

Weber, B., et al., 2008. Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. *Journal on Data and Knowledge Engineering*, 66 (3), pp. 438–466.

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

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