# Context-Aware Adaptive Process Information Systems: The Context-BPMN4V Meta-Model

Imen Ben Said, Mohamed Amine Chaabane, Éric Andonoff, Rafiq Bouaziz

# Context-Aware Adaptive Process Information Systems: The Context-BPMN4V Meta-Model

Imen Ben Said[1], Mohamed Amine Chaabane[1], Eric Andonoff[2], and Rafik Bouaziz[1]

[1] MIRACL/FSEG, Route de l'aéroport, BP 1088, 3018 Sfax, Tunisia
{Imen.Bensaid,MA.Chaabane,Raf.Bouaziz}@fsegs.rnu.tn
[2] IRIT/UT1-Capitole, 2 rue du doyen Gabriel Marty, 31042 Toulouse, France
andonoff@univ-tlse1.fr

**Abstract.** This paper introduces Context-BPMN4V, an extension of BPMN for modeling variability of processes using versions and also considering their contextual dimension. More precisely, it shows how we extend BPMN meta-models to support version modeling to deal with process adaptation, along with context modeling to characterize the situations in which instances of processes are executed. Because of space limitation, this paper only focuses on intra-organizational processes.

**Keywords:** Context, Adaption, Process Information Systems, Versions, BPMN.

## 1 Introduction

In the last two decades, there has been a shift from data-aware information systems to Process-Aware Information Systems [1]: processes play now a fundamental role in Enterprise Information Systems (EIS) and they are the support of the alignment between EIS and business strategies of enterprise actors [2,3]. A Process (aware) Information System (PIS) is a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process schemas (models). Examples of PISs are workflow management systems (*e.g.* YAWL), case-handling systems (*e.g.* Flowers), enterprise resource planning systems, Business Process Management (BPM) suites *(e.g.* BizAgi) and service oriented architecture-based process implementation [1].

Adaptation is a major challenge for PIS, before their definitive acceptance and use in enterprises [4]. This issue is fundamental, as the economic environment in which enterprises are involved is more and more dynamic, competitive, and open [5,6]: enterprises frequently change their processes in order to meet, as quickly and efficiency as possible, new operational, organizational or customer requirements, or new law regulations. Thus, the economic success of enterprises is closely related to their ability to integrate changes happening in their environment and to make evolve their processes accordingly [7].

This need for adaptive PIS has led BPM researchers to intensively investigate process adaptation issue. Several typologies have been introduced to classify process

adaptation, and even if they are different, they all agree to distinguish two different times for process adaptation (design-time and run-time), two abstraction levels for process adaptation (the schema –model– level and the instance –case– level), and three types of process adaptation [8–11]: (i) *adaptation by design*, for handling foreseen changes in processes where strategies are not necessarily defined at design-time to face these changes and must be specified at run-time by process users (*e.g.* late modelling and late biding [12]), (ii) a*daptation by deviation*, for handling occasional unforeseen changes and where the differences with initial process are minimal, and (iii) *adaptation by evolution*, for handling unforeseen changes in processes, which require occasional or permanent modifications in process schemas.

We distinguish several approaches to deal with process adaptation issue: activity-driven approach [13–21], constraint-driven approach [22,23], data-driven approach [24,25], case-driven approach (case handling) [26], and more recently, social-driven approach [27]. Activity-driven approach is based on the explicit representation of process schemas (models): the activities of the process and the way there are synchronized are modelled along with information handled and resources involved in it. On the other hand, constraint-driven, case-driven, social-driven and even date-driven approaches avoid describing the way activities are synchronized: in these approaches, process schemas are not explicitly and a priori defined. Even if these approaches are promising, this paper focuses on activity-driven adaptive PIS, as activity-oriented models are used in the majority of (service-oriented) process management systems. Consequently, BPM community has to provide solutions to deal with activity-oriented process adaptation [7].

When dealing with adaptation in activity-driven PIS, in addition to the behavioural, organisational and informational dimensions usually considered for processes and which respectively define the activities of the process and their synchronization, the involved resources in the realization of these activities and the data they produce or consume, we also have to consider the contextual dimension of processes in order to characterize the situations in which instances of processes are executed [7,28–30].

This paper focuses on context-aware adaptation in activity-driven PIS. More precisely, it presents Context-BPMN4V, an extension of BPMN to model context of versions of processes. Versions have been introduced in activity-driven PIS to deal with adaptation issue as they facilitate adaptation by design, by deviation and by evolution [7]. They also facilitate the migration of process instances from an initial schema to a final one, allowing cases in which this migration is impossible and enabling the execution of a same process according to different schemas [11,13–16]. BPMN4V (BPMN for Versions) is an extension of BPMN to model process variability using versions. Basic concepts for process versions have been introduced in [7,21] and BPMN4V has been presented in [31]. In this paper, we extend BPMN4V in order to consider context of process versions. Because of space limitation, we particularly focus on intra-organizational processes [32].

This paper is organized as follows. Section 2 focuses on related works about adaptation in activity-driven PIS and context in BPM area. Section 3 presents BPMN4V to deal with process adaptation using version of processes. Section 4 shows how we extend BPMN4V in order to take into account the contextual dimension of processes. Finally, section 5 concludes the paper and gives some directions for future works.

## 2 Related Works

Adaptation in activity-driven PIS is a highly investigated issue since the end of the nineties. Even if existing contributions are significant [12–21], considering run-time and design-time adaptation, both at the schema and the instance level, we can observe that they mainly focus on adaptation of process behavior, leaving aside the organizational, informational and contextual dimensions of processes. Moreover, proposed notations are not standards and are unlikely to be used by process designers, who are in charge of modeling variability of processes.

In order to take into account the second previous remark, some contributions have extended BPMN, which is now known as the standard notation for processes, to deal with adaptation issue. For instance, [33] has extended BPMN including concepts of a generic meta-model to support process goals and process performance. [34] has investigated more deeply this issue, proposing goals models, depicted as tree-graphs, to represent process goals and their possible variations. However, these works did not adopt a comprehensive approach, considering all the dimensions of processes at the same time, *i.e.* the behavioral, organizational, informational and contextual dimensions of processes.

The notion version has been recognized as a key notion to deal with adaptation issue. On the one hand, handling version of processes facilitate the migration of instances from an initial schema to a final one, allowing if the migration is not possible, two different instances of a same process to run according to two different schemas [7,13–16,21]. On the other hand, as defended in [7], versions are appropriate to deal with the three types of adaptations identified in the main topologies of the literature ([8–11]), *i.e.* adaptation by design, adaptation by deviation and adaptation by evolution. Consequently, in a previous proposition, and with respect to the concepts of [21], we introduced BPMN4V (BPMN for Versions), an extension of BPMN to support intra-organizational and inter-organizational process version modeling, both considering behavioral (what, how), organizational (who) and informational (when) dimensions of processes [31]. Thus, we extended in this proposition the main contributions about versions in BPM [13–18,20]. We continue our effort in adaptive in activity-driven PIS area extending BPMN4V to integrate the contextual dimension of processes in order to characterize the situations in which instances of processes are executed and thus to define *why* version of processes are defined/used instead another according to the context.

Context-awareness has been investigated in several domains of computer science (*e.g.* natural language, human computer interaction, mobile application, or web system engineering). BPM area has also taken advantages from context-awareness. A process context is defined in [28] as *the minimum set of variables containing all relevant information that impact the design and execution of a process*. This contribution also introduced a taxonomy for contextual information by distinguishing four natures of context: (i) *immediate context*, which covers information on process components, *i.e.* context of activities, events, control flow, data and organization, (ii) *internal context*, which covers information on the internal environment of an organization that impacts the process (*e.g.* process goals, organization strategies and

policies), (iii) *external context*, which covers information related to external stakeholders (*e.g.* customers, suppliers, government) of the organization, and finally (iv) *environmental context*, which covers information related to external factors (*e.g.*, weather, time, workforce, economic data). Other works (*e.g.* [21], [34]) only distinguish two types of information to represent process context: functional information related to process components (activities, events, resources…) and non-functional information related to quality of process (safety, security, cost, time…). In this work, we rather adopt the taxonomy introduced in [28], which is more comprehensive.

In addition to these taxonomies, several contributions have been done to support context awareness in process modeling. For instance, [29,30,35] proposed a rule-based approach to define contextual information used to configure process instances to particular situations. On the other hand, [34,36] proposed goals models, depicted as tree-graph, to represent process goals and their possible variations. In these works, a goal model is used to adapt process definition according to the context. But only considering goal is not enough to describe context of processes. Finally, [21] also investigated the contextual dimension of processes, but as indicated in the previous section, the underlying context taxonomy is rather incomplete, and the proposed notation is unlikely to be used by process designers.

To sum up, in this paper we advocate the modeling of process versions using BPMN4V (BPMN for Versions) to deal with adaptation in activity-driven PIS [31]. We also define Context-BPMN4V, an extension of BPMN4V in order to take into account context for (process) versions and considering the taxonomy introduced in [28]. This contextual dimension is fundamental to help PIS users (i) at design-time, to indicate why a process version is defined (different variables and conditions are specified for the considered process version), and (ii) at run-time, to instantiate a particular version of a process according to a concrete situation, *i.e.* a set of values given to the different variables specified in the context of the considered process version.

# 3 Modeling Process Versions: The BPMN4V Meta-Model

As indicated before, this paper focuses on intra-organizational processes. Such processes are modelled in BPMN2.0 through private processes, which are internal to a specific organization.

This section first introduces the notion of version and then present BPMN4V, an extension of BPMN for modeling versions of private process.

## 3.1 Notion of Version

As illustrated in Fig. 1 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 changes 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.

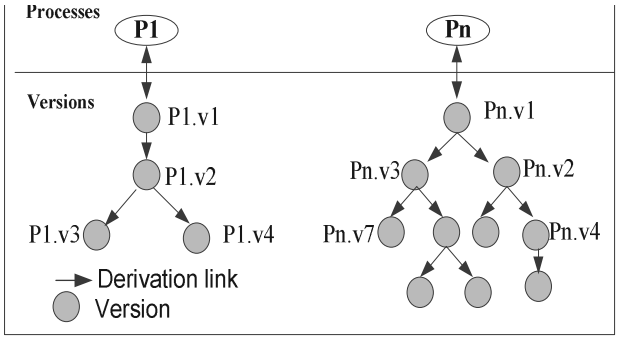


**Fig. 1.** Versions to Model Process Variability

We defend that this version notion subsumes the notion of variant as it is defined in [17,19,20,29] as, when considering versions, we both model process evolution and process alternative (*i.e.* variant) to describe process variability and deal with adaptation by design, adaptation by deviation and adaptation by evolution [7].

### 3.2 BPMN4V: BPMN2.0 for Versions

BPMN4V meta-model for private processes results from the merging of BPMN2.0 meta-model for private processes and a versioning pattern used to make classes of BPMN2.0 meta-model versionnable, *i.e.* able to handle versions. We present briefly these two layers.

**Versioning Pattern.** The versioning pattern 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 Fig. 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*.

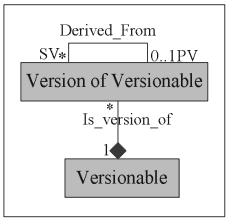


**Fig. 2.** Versioning Pattern

The *Is_version_of* relationship links a class to its corresponding versions. The *Derived_From* relationship allows for building version derivation hierarchies (cf. Fig. 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.

**Main Concepts of BPMN 2.0 Meta-Model [37].** These concepts are presented in Fig. 3 hereafter. The three main dimensions of processes are considered in BPMN2.0 meta-model.

The behavioral dimension of processes supports the description of process activities and their synchronization along with events happening during process execution through the notions of *FlowElementContainer* which contains *SequenceFlow*, *FlowNode* (*Gateway*, *Event*, and *Activity*), and *Data Object*. A *SequenceFlow* is used to show the order of *FlowNode* in a process. A *SequenceFlow* may refer to an *Expression* that acts as a gating condition. The *Expression* class is used to specify a condition using natural-language text. 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 it reacts to something (*catchEvent*), or it can throw a result (*throwEvent*). An *Event* can be composed of one or more *EventDefinitions*. There are many types of *Event Definitions*: *ConditionalEventDefinition, TimerEventDefinition…* 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). BPMN2.0 identifies different types of tasks: *Service Task*, *User Task*, *Manual Task*, *Send Task* and *Receive Task*.

Regarding the organizational dimension of processes, an activity is accomplished 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*.

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*.

**BPMN4V Meta-Model.** The idea is to use the versioning pattern introduced before (cf. Fig. 2) to make some classes of the BPMN2.0 meta-model versionable, i.e. able to handle versions. Fig. 3 below presents the resulting meta-model. White rectangles and relationships correspond to classes and relationships of BPMN2.0, while grey rectangles and blue relationships correspond to classes and relationships involving versions, and resulting from the introduction of the versioning pattern.

We propose to handle versions for seven classes: *Process*, *Sub Process*, *Event*, *Activity*, *ItemAwareElement, Resource*, and *ResourceRole* in order to support adaptive

activity-driven PIS. Different instances (versions) of these classes can be created, each one representing a significant state (alternative 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 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. The general idea is to keep track of changes occurring to components participating to the description of the way business is carried out.

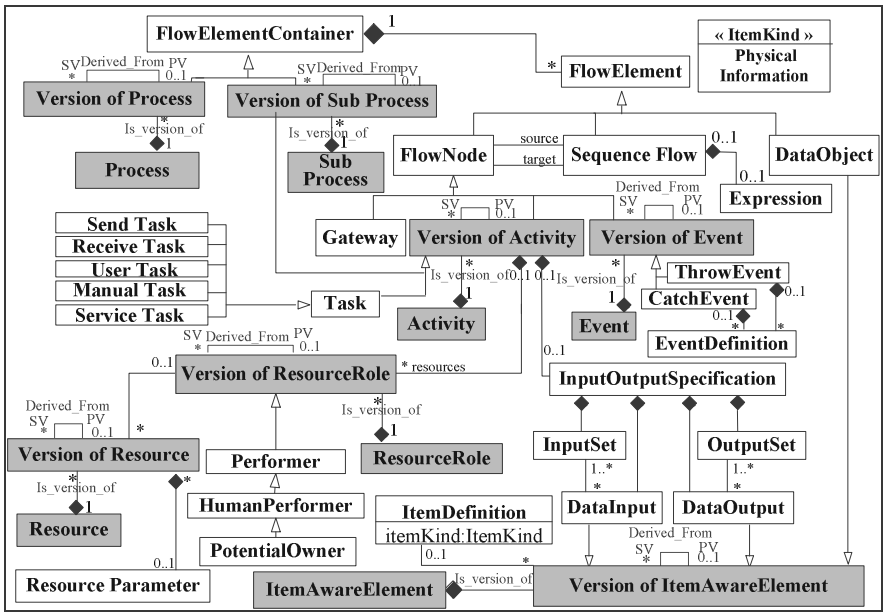


**Fig. 3.** BPMN4V Meta-model for Modeling Process Versions

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 [21]. We also propose to create versions of *ResourceRole* 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 *ResourceRole* 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 data (*Itemkind* is a *Physical* data), and if after technical changes it becomes an electronic data (*Itemkind* becomes an *Information* data), then a new version of *Report* has to be created.

Regarding the behavioral dimension of processes, several classes are versionable: *Event*, *Activity*, *Sub-Process* and, of course, *Process*. More precisely regarding

activities, we create a new 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 and events or in the way they are linked together (used patterns, *i.e.* gateways, are changed).

### 3.3 Example

We illustrate in Fig. 5 the instantiation of BPMN4V meta-model according to the damage compensation process of an insurance company. This process is shown in Fig. 4 below. Basically, it contains five activities: *Receive Request*, *Review Request*, *Send reject letter*, *Calculate claim amount*, and *Financial settlement*. For simplification reasons, we only focus on the behavioral dimension of this process.

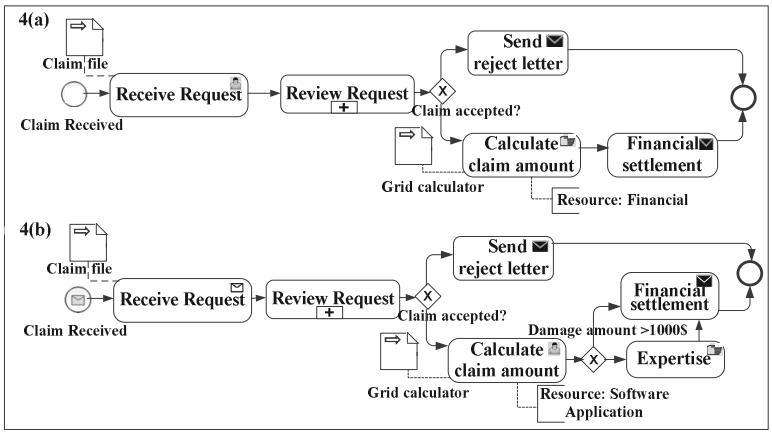


**Fig. 4.** Versions of the Damage Compensation Process

Two versions of this process are described in Fig. 4. The first one is given in Fig. 4(a). This version starts when the client files a claim. After checking the claim, a reject letter is sent, by mail, if the request is not accepted. Otherwise, the claim amount is calculated by the insurance manager using *GridCalculator*, and the financial service prepares and sends the financial settlement. On the other hand, further to an increase in the number of its customers, the insurance agency has modeled a second version of this process. Fig. 4(b) illustrates this version introducing an *Expertise* activity (a new activity, used when the damage amount exceeds 1000$) and both modifying the start *ClaimReceived* event and the *ReceiveRequest* and *CalculateClaimAmount* activities (their type have changed). To sum up, regarding the damage compensation process we have two versions of the process itself, two versions of *ClaimReceived* event and two versions of the *ReceiveRequest* and *CalculateClaimAmount* activities: the first version of both *ReceiveRequest* and *CalculateClaimAmount* activity hold for the first version of the process while the

second ones hold for the second version of the process. In addition, the sequence flows and patterns have been modified in the second version of the process. Finally, regarding the *ClaimReceived* event, in the first version of the process, it is a *None Event*, used to indicate that this version starts when the client presents its *ClaimFile* (a paper data). However, in the second process version, it becomes a *Message Event*, indicating that the client sends the *ClaimFile* (an electronic data) as a message via the insurance web site.

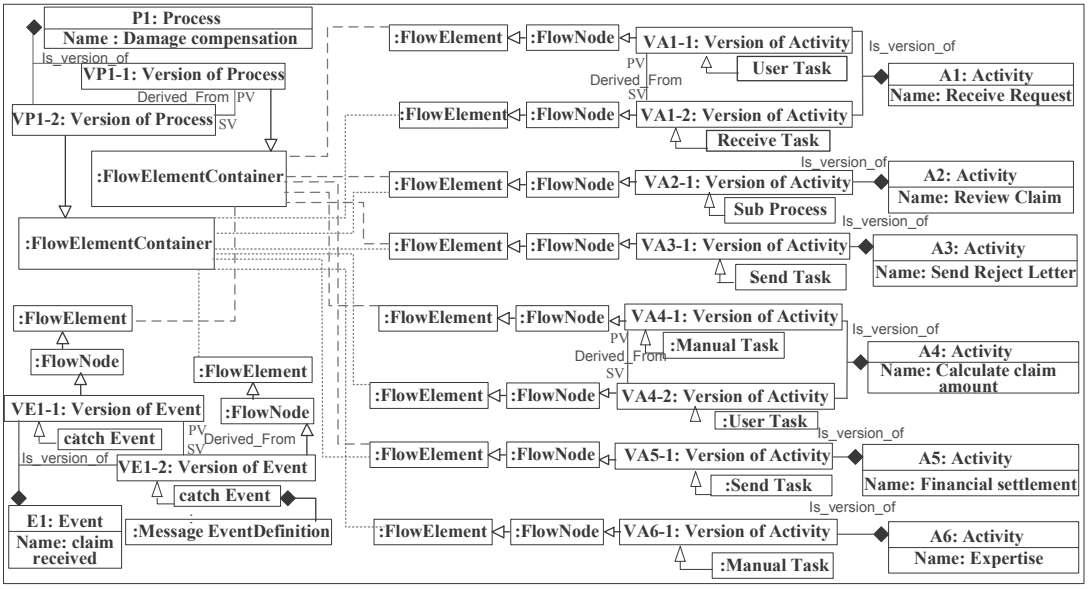


**Fig. 5.** BPMN4V Instantiation for the Damage Compensation Process

# 4 Considering Context in BPMN4V: The Context-BPMN4V Meta-Model

We advocate using the version notion to support adaptation in activity-driven PIS. Thus, several versions can be defined for a process: versions of the process itself, but also versions of activities, sub-processes, events, data produced or consumed by activities (*i.e.* ItemAwareElement) and versions for the organizational dimension of processes (*i.e.* Resource and ResourceRole). Each of the defined versions is required in a specific context, *i.e.* has to be used in a given situation. Therefore, it becomes crucial to consider the contextual dimension of versions in order to characterize the situations in which these versions have to be used. Indeed, this contextual dimension is fundamental to help PIS users (i) at design-time, to indicate why a (process) version is defined –different variables and conditions are specified for the considered (process) version–, and (ii) at run-time, to instantiate a particular version of a process according to a concrete situation, *i.e.* a set of values given to the different variables specified in the context of the considered process version.

We present below Context-BPMN4V, an extension of BPMN4V considering the contextual dimension of processes. We first introduce a Context meta-model for context description in PIS. Then, we present Context-BPMN4V which results from

the merging of this Context meta-model and BPMN4V to model context for (process) versions. Finally, we illustrate context definition for process versions using the damage compensation process introduced previously.

### 4.1 Context Modeling

The Context meta-model given in Fig. 6 below allows the definition of a *Context Model* as the aggregation of a set of context parameters. A *Context Parameter* corresponds to a variable characterizing a situation, and to which a condition will be defined. A context parameter has a *Context Nature*, which can be immediate, internal, external or environmental, according to the taxonomy given in [28]. This taxonomy is used to specify the source of each parameter that composes a *Context Model.* In addition to this taxonomy, we also consider the type of a *Context Parameter*, which refers to the dimension to which it belongs to. Thus, we consider *Behavioral Parameters*, *i.e.* variables related to the behavioral dimension of processes (*e.g.* activity execution mode, activity duration), *Role Parameters*, *i.e.* variables related to the organizational dimension of processes (*e.g.* availability of a resource, experience of a human performer), and *Data Parameters*, *i.e.* variables related to the informational dimension of processes (*e.g.* data type, data structure). We finally consider *Goal Parameters*, *i.e.* variables describing objectives to be achieved (*e.g.* quality, cost, quantity); such type of context parameters belongs to the intentional dimension of processes [35,38].

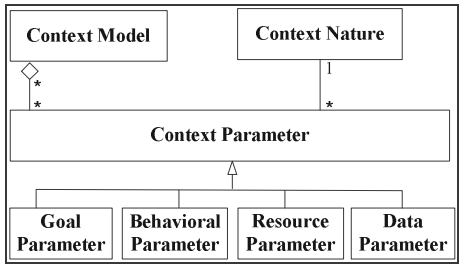


**Fig. 6.** Context Meta-model

### 4.2 Extending BPMN4V Meta-Model to Consider Context for (Process) Versions: The Context-BPMN4V Meta-Model

Context-BPMN4V, visualized in Fig. 7, results from the merging of BPMN4V and Context meta-models. It defines the necessary concepts for modeling context of process versions. Of course, context can be defined for each versionable component of the process; thus, in addition to process versions, contexts can be defined for versions of activities, sub processes, events, resource roles, resources and ItemAwareElements.

The proposed meta-model links a process to a context model aggregating a set of context parameters, corresponding to variables from the different dimensions of the considered process: goal parameters from the intentional dimension of the process, behavior parameters from the behavioral dimension of the process, resource

parameters from the organizational dimension of the process, and data parameters from the informational dimension of the process. Thus, each versionable component of the process can be linked to one or several context parameters of its corresponding context model, in order to define conditions on these parameters.

More precisely, goal parameters specify objectives of versionable concepts through goal conditions. A *Goal Condition* is a boolean expression defining why a version is created (cf. section 4.3 for an goal condition example). As indicated before, each versionable component of a process can be linked to a parameter of its corresponding context model. As a consequence, a goal parameter, used in the definition of goals of a versionable component of a process, has to be part of the set of context parameters that form the context model of this process. In the proposed meta-model, we also use OCL constraints to define restrictions. We give a textual definition of these restrictions in Fig. 7.

Regarding behavioral, data and role parameters, we specify conditions, called *Assignment conditions* that allow the assignment of versionable components. These conditions, described as boolean expressions, define for a specific version of process, the situation in which versions of activities, of events, of resource role and of ItemAwareElement involved in this process have to be used.

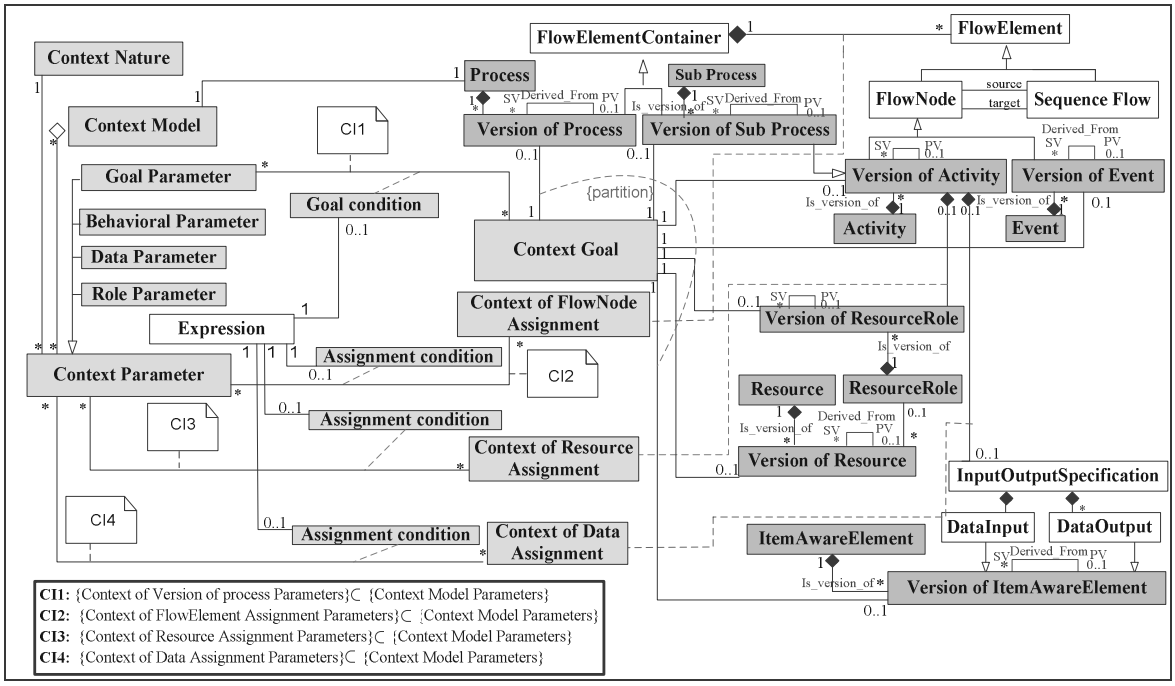


**Fig. 7.** Context-BPMN4V Meta-model

Context-BPMN4V extends BPMN4V adding three classes used to define assignment conditions within relationships between BPMN components:

- *Context of FlowNode Assignment* attached to the relationship between *FlowElementContainer* and *FlowElement*, defines conditions indicating in which situation a version of activity (or a version of event) has to be used in a version of process or a version of sub-process.

- In the same vein, *Context of Resource Assignment*, attached to the relationship between *Version of Activity* and *Version of ResourceRole*, defines conditions indicating in which situation a version of activity is performed by a version of *ResourceRole*.
- Finally, *Context of Data Assignment*, attached to the relationship between *Version of Activity* and *InputOutputSpecification*, defines conditions indicating in which situation a version of activity consumes or produces versions of *ItemAwareElement*.

Finally, a context parameter used in the definition of an assignment condition must belong to the set of context parameters that form the context model of the corresponding process. Constraints *CI2*, *CI3* and *CI4* express these restrictions with respect to *Context of FlowNode Assignment*, *Context of Resource Assignment* and *Context of Data Assignment* classes.

### 4.3 Defining Context in the Damage Compensation Process

To better illustrate our proposition, we refer to the damage compensation process previously presented (cf. section 3.3), and gives the context of each version of this process. Even if this example is rather simple, it illustrates suitably this context notion for processes. Table1 below summarizes context parameters involved in the first and the second version context definition of this process.

**Table 1.** Context Parameters and Conditions in the Damage Compensation Process

| Context Nature | Parameter/ (Parameter type) | Context of the first version | Context of the second version |
|---|---|---|---|
| External Context | NumberofDailyClaims / (Goal parameter) | <50 | ≥50 |
| Immediate Context | ClaimFile / (Data parameter) | is-a paperData | is-a ElectronicData |
| | GridCalculator / (Data parameter) | is-a paperData | is-a ElectronicData |
| | CalculateClaimAmount / (Behavioral parameter) | is-a ManualActivity | is-a ElectronicActivity |
| | CalculateClaimAmountRole / (Role parameter) | is-performed-by Human | is-performed-by ACA |
| | Expertise / (Behavioral parameter) | | is-a ManualActivity |

The first version of the damage compensation process is defined at the beginning, *i.e.* when insurance agency is created: only few clients and thus only few daily claims, rather simple, to deal with, and only few IT investments. In this process version, the insurance manager calculates the claim amount using a grid calculator. Parameters

that characterize this situation are the number of daily claims to deal with, the claim file modeled as a paper data, and the way the claim amount is calculated: manually from a grid calculator. The second version of the process is defined to face the increasing number of clients, and thus the increasing number of daily claims to deal with. The insurance agency invested in IT software: clients no longer fill in claim files manually but rather declare their claims via the website of the insurance agency, and specific software is used to calculate claim amounts. In addition, a new role and a new activity are introduced in the process when specific claim require expertise. Parameters that characterize this second version of the damage compensation process are the number of daily claims, the claim file which is modeled as an electronic data, the presence of the expertise activity, and the way claim amounts are calculated: automatically using specific software, or manually using grid calculator.

More precisely, *NumberofDailyClaims* is a goal parameter whose nature is external: the two conditions *NumberofDailyClaims<50* and *NumberofDailyClaims>=50* define the possible values for this parameter in the two versions of the damage compensation process. *ClaimFile* and *GridCalculator* are immediate data parameters. The conditions, *is-a paperData* and *is-a ElectronicData*, define the possible values for these parameters in the two process versions. Note that *ClaimFile* and *GridCalculator* refer to data handled by the process activities. In the same vein, we refer to existing activities with the immediate behavioral parameters *CalculateClaimAmount* and *Expertise*. Regarding this latter, it is not involved in the first version of the process but it is a manual activity in the second version of the process (*is-a ManualActivity* condition). Regarding *CalculateClaimAmount* parameter, two conditions are defined, respectively in the context of the first and the second process versions: it is a manual activity in the first version (*is-a ManualActivity*) while it is an electronic activity in the second version (*is-a ElectronicActivity*). Finally, immediate resource parameters are defined for both *CalculateClaimAmount* and *Expertise* activities in order to define how these activities are performed. Thus, *CalculateClaimAmountRole* is performed by a human in the first process version (*is-performed-by Human*), while it is performed by a software application in the second process version (*is-performed-by ACASoftwareApplication*). In the same way, *ExpertiseRole* is performed by a human in the second process version. Note that classical and specific operators are used to specify these conditions (*e.g.* is-a, is-performed-by, has-experience) [21]. Fig. 8 below gives an extract of the instantiation of damage compensation process context.

In this figure, *CG1* and *CG2* are context goals (*i.e.* instances of the ContextGoal class) of the first and the second version of this process (*VP1-1* and *VP1-2*). These goals are defined using *NumberofDailyClaims* context parameter and <50 (Gc1) and >=50 (Gc2) goal conditions. In addition, *CFA1*, *CFA2*, *CDA1, CDA2, CRA1* and *CRA2* define why a version of the calculate claim amount activity is used in the two damage compensation process versions. Actually, two versions of this activity are created: *VA4-1* and *VA4-2*, each involved in one of the two process versions. *VA4-1* holds for the first version of this process: in this version, *CalculateClaimAmount* is a *ManualActivity*, *CalculateClaimAmountRole* is performed by *Human*, and *GridCalculator* is a *paperData*. Regarding *VA4-2*, it holds for the second version of the damage compensation process: in this case, *CalculateClaimAmount* is a *ElectronicActivity*, *CalculateClaimAmountRole* is performed by *ACA*, and *GridCalculator* is an *electonicData*.

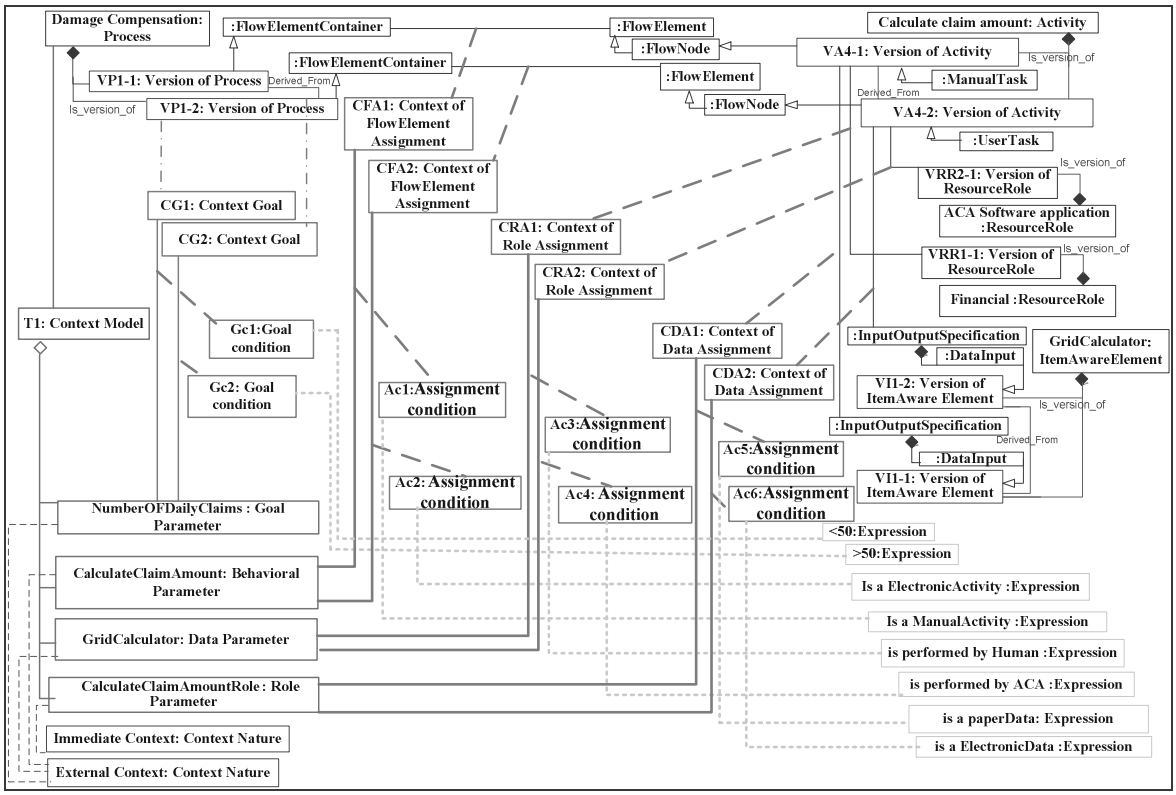


**Fig. 8.** Extract of the damage compensation process context instantiation

# 5 Conclusion

This paper has presented Context-BPMN4V, an extended BPMN meta-model to support context-aware process variability modeling in activity-driven PIS. It results from the merging of BPMN4V and a Context meta-model. Context-BPMN4V supports the modeling of adaptive processes using versions. Indeed, versions are a powerful mechanism to model process variability: (i) they facilitate the migration of process instances from an initial schema to a final one, allowing cases in which this migration is impossible and enabling the execution of a same process according to different schemas, and (ii) they are appropriate to deal with the three types of adaptation identified in the different typologies of literature: adaptation by design, adaptation by deviation and adaptation by evolution.

In addition, Context-BPMN4V also considers the contextual dimension of processes, which rather focuses on the why dimension of a process. This dimension is considered for each versionable component of Context-BPMN4V: thus, context can be defined for versions of processes of course, but also for versions of activities, sub-processes, events, resource roles, and data consumed and produced by activities or sub-processes. This contextual dimension is fundamental to help PIS users (i) at design-time, to indicate why a process version is defined (different variables and conditions are specified for the considered process version), and (ii) at run-time, to instantiate a particular version of a process according to a concrete situation, *i.e.* a set of values given to the different variables specified in the context of the considered process version.

The advantages of our contribution are the following. First, it extends BPMN which is a standard notation: it makes our proposition potentially to be used by process designers. Second, regarding versions, it extends the different contributions of literature considering the notion of version for both dimensions of processes: behavioral, organizational, informational, and contextual in order to define in which situation a given (process) version has to be used.

Our future works will take two directions. First, we will define a language to help PIS users at run-time, to instantiate a particular version of a process according to a concrete situation: this language must support the matching between current concrete situations and contexts defined for process versions. Second, we will implement Context-BPMN4V in order to provide PIS users with a specific tool for context-aware and adaptive intra and inter organizational processes. The consequence will be the introduction of a specific graphical notation for versions and their context.

## References


1. Dumas, M., van der Aalst, W., ter Hofstede, A.: Process-Aware Information Systems: Bridging People and Software through Process Technology. Wiley (2005)
2. Rolland, C.: Fitting System Functionality to Business Needs: Alignment Issues and Challenges. In: International Conference on Software Methodologies, Tools and Techniques, Yokohama City, Japan, pp. 137–147 (September 2010)
3. Simonin, J., Nurcan, S., Barrios, J.: Evolution organisationnelle fondée sur la cohérence des relations entre acteurs avec les buts métier. In: National Conference on Informatique des Organisations et des Systèmes d'Information et de Décision, Paris, pp. 225–240 (May 2013)
4. Smith, H., Fingar, P.: Business Process Management: the Third Wave. Megan-Kiffer Press (2003)
5. Reijers, H.: Workflow Flexibility: the Forlon promise International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Manchester, United Kingdom, pp. 271–272 (June 2006)
6. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer (2007)
7. Chaâbane, M.A., Andonoff, E., Bouaziz, R., Bouzguenda, L.: Versions to Address Business Process Flexibility Issue. In: Grundspenkis, J., Morzy, T., Vossen, G. (eds.) ADBIS 2009. LNCS, vol. 5739, pp. 2–14. Springer, Heidelberg (2009)
8. Nurcan, S.: A Survey on the flexibility Requirements related to Business Process and Modelling Artifacts. In: International Conference on System Sciences, Waikoloa, Big island, Hawaii, USA, pp. 378–387 (January 2008)
9. Schonenberg, H., Mans, R., Russel, N., Mulyar, N., van der Aalst, W.: Process Flexibility: a Survey of Contemporary Approaches. In: Dietz, J.L.G., Albani, A., Barjis, J. (eds.) CIAO! 2008 and EOMAS 2008. LNBIP, vol. 10, pp. 16–30. Springer, Heidelberg (2008)
10. Weber, B., Sadiq, S., Reichert, M.: Beyond Rigidity – Dynamic Process Lifecycle Support: a Survey on Dynamic Changes in Process-Aware Information Systems. International Journal on Computer Science, Research and Development 23(2), 47–65 (2009)
11. Andonoff, E., Nurcan, S., Hanachi, C.: Adaptation des processus d'entreprise. In: Lopisteguy, P., Rieu, D., Roose, P. (eds.) L'adaptation dans tous ses états, ch. 3, Cepadues (2012)

12. Adams, M., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Worklets: a Service-Oriented Implementation of Dynamic Flexibility in Workflows. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 291–308. Springer, Heidelberg (2006)
13. Casati, F., Ceri, S., Pernici, B., Pozzi, G.: Workflow Evolution. In: Thalheim, B. (ed.) ER 1996. LNCS, vol. 1157, pp. 438–455. Springer, Heidelberg (1996)
14. Kammer, P., Bolcer, G., Taylor, R., Bergman, R.: Techniques for supporting dynamic and adaptive workflow. International Journal on Computer Supported Cooperative Work 9(3/4), 269–292 (1999)
15. Kradolfer, M., Geppert, A.: Dynamic workflow schema evolution based on workflow type versioning and workflow migration. In: International Conference on Cooperative Information Systems, Edinburgh, Scotland, pp. 104–114 (September 1999)
16. Reichert, M., Rinderle, S., Dadam, P.: ADEPT workflow management system: flexible support for enterprise-wide business processes. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 370–379. Springer, Heidelberg (2003)
17. Lu, R., Sadiq, S.K.: Managing process variants as an information resource. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 426–431. Springer, Heidelberg (2006)
18. Zhao, X., Liu, C.: Version Management in the Business Change Context. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 198–213. Springer, Heidelberg (2007)
19. Lu, R., Sadiq, S., Governatori, G., Yang, X.: Defining adaptation constraints for business process variants. In: Abramowicz, W. (ed.) BIS 2009. LNBIP, vol. 21, pp. 145–156. Springer, Heidelberg (2009)
20. Hallerbach, A., Bauer, T., Reichert, M.: Capturing Variability in Business Process Models: the Provop Approach. Journal of Software Maintenance 22(6-7), 519–546 (2010)
21. Chaâbane, M., Andonoff, E., Bouaziz, R., Bouzguenda, L.: Modélisation multidimensionnelle des versions de processus. Journal on Ingénierie des Systèmes d'Information 15(5), 89–114 (2010)
22. Pesic, M., Schonenberg, H., Sidorova, N., van der Aalst, W.: DECLARE: full support for Loosely-Structured Processes. In: International Conference on Enterprise Distributed Object Computing, Annapolis, Maryland, USA, pp. 287–300 (October 2007)
23. Pesic, M., Schonenberg, M.H., Sidorova, N., van der Aalst, W.M.P.: Constraint-based workflow models: Change made easy. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 77–94. Springer, Heidelberg (2007)
24. Müller, D., Reichert, M., Herbst, J.: Data-driven Modeling and Coordination of Large Process Structures. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 131–149. Springer, Heidelberg (2007)
25. Müller, D., Reichert, M., Herbst, J.: A New Paradigm for the Enactment and Dynamic Adaptation of Data-Driven Process Structures. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 48–63. Springer, Heidelberg (2008)
26. van der Aalst, W., Weske, M., Grünbaur, D.: Case Handling: a New Paradigm for Business Process Support. International Journal on Data Knowledge Engineering 53(2), 129–162 (2005)
27. Bruno, G., Dengler, F., Jennings, B., Khalaf, R., Nurcan, S., Prilla, M., Sarini, M., Schmidt, R., Silva, R.: Key challenges for enabling agile BPM with social software. Journal of Software Maintenance and Evolution: Research and Practice 23(4), 297–326 (2011)

28. Rosemann, M., Recker, J.: Context-aware Process Design Exploring the Extrinsic Drivers for Process Flexibility. In: CAiSE Conference on Business Process Modeling, Development and Support, Luxembourg (June 2006)
29. Hallerbach, A., Bauer, T., Reichert, M.: Context-based Configuration of Process Variants.In: ICEIS Workshop on Technologies for Context-Aware Business Process Management, Barcelona, Spain (June 2008)
30. Saidani, O., Nurcan, S.: Context-Awareness for Adequate Business Process Modelling. In: International Conference on Research Challenges in Information Science, Fés, Morocco, pp. 177–186 ( May 2009)
31. Ben Said, I., Chaabane, M., Andonoff, E., Bouaziz, R.: Extending BPMN 2.0 meta-models for Process Version Modelling. In: International Conference on Enterprise Information Systems, Lisbon, Portugal (April 2014)
32. Divitini, M., Hanachi, C., Sibertin-Blanc, C.: Inter Organizational Workflows for Enterprise Coordination. In: Omicini, A., Zambonelli, F., Klusch, M., Tolksdorf, R. (eds.) Coordination of Internet Agents, pp. 46–77. Springer (2001)
33. Korherr, B., List, B.: Extending the EPC and the BPMN with Business Process Goals and Performance Measures. In: International Conference on Enterprise Information Systems, Funchal, Madeira, Portugal (June 2007)
34. Santos, E., Pimentel, J., Castro, J., Sánchez, J., Pastor, O.: Configuring the variability of business process models using non-functional requirements. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) BPMDS 2010 and EMMSAD 2010. LNBIP, vol. 50, pp. 274–286. Springer, Heidelberg (2010)
35. Saidani, O., Nurcan, S.: Towards Situational Business Process Modelling. In: CAiSE Forum, Montpellier, France, pp. 93–96 (June 2008)
36. Liaskos, S., Lapouchnian, A., Yu, Y., Yu, E., Mylopoulos, J.: On Goal-based Variability Acquisition and Analysis. In: International Conference on Requirements Engineering, Minneapolis/St.Paul, Minnesota, USA, pp. 76–85 (September 2006)
37. OMG,. Business Process Model and Notation (BPMN) Version 2.0. OMG Document Number: formal/2011-01-03 (2011), http://www.omg.org/spec/BPMN/2.0.
38. Nurcan, S., Edme, M.: Intention Driven Modelling for Flexible Workflow Applications. International Journal on Software Process: Improvement and Practice 10(4), 363–377 (2005)