A context-based approach for modelling and querying versions of BPMN processes

Mohamed Amine Chaâbane*, Imen Ben Said, Fatma Ellouze and Rafik Bouaziz

MIRACL,
Université de Sfax,
Route de l'aéroport, BP 1088, 3018, Sfax, Tunisia
Email: mohamedamine.chaabane@isaas.usf.tn
Email: Imen.bensaid@fsegs.usf.tn
Email: fatma.ellouze@enis.usf.tn
Email: rafik.bouaziz@usf.tn
*Corresponding author

Eric Andonoff

IRIT,
Université Toulouse 1 – Capitole,
2 rue du Doyen Gabriel Marty,
31042, Toulouse Cedex, France
Email: andonoff@univ-tlse1.fr

Abstract: Versioning is an interesting solution to deal with business process flexibility. It consists of the definition of several process versions for taking into account the significant changes occurring to the processes. It allows the running of several instances of the same process according to different models. However, in a multi-version environment, where numerous versions co-exist, it is important to specify the context in which these versions can be used. The context is used in particular to find out, for a given situation, the appropriate process version to be executed at run-time. We aim in this paper at offering a solution to model the context of versions of intra- and inter-organisational processes and query these versions using their context. More precisely the recommended solution extends BPMN2.0, the de-facto standard for process modelling, to consider versions and contexts, and introduces a context based language for versions querying.

Keywords: business process flexibilty; modelling versions; BPMN private processes; BPMN collaborations; BPMN for versions.

Reference to this paper should be made as follows: Chaâbane, M.A., Said, I.B., Ellouze, F., Bouaziz, R. and Andonoff, E. (2020) 'A context-based approach for modelling and querying versions of BPMN processes', *Int. J. Business Process Integration and Management*, Vol. 10, No. 1, pp.62–86.

Biographical notes: Mohamed Amine Chaâbane received his PhD from the University of Toulouse, France in 2012. He is an Associate Professor at the Higher Institute of Business Administration, University of Sfax, Tunisia. Since 2017, he was the Head of the Computer Science and Quantitative Methods Department. He is also a Member of the Multimedia, InfoRmation Systems and Advanced Computing Laboratory (MIRACL). His research interests focus on business process management field. He is working on topics related to business process modelling, process flexibility, context of process and self-adaptation of process.

Imen Ben Said received her PhD in Computer Science from the University of Sfax and University of Toulouse 1 Capitole in 2017. Since 2009, she was a teacher/researcher at the Faculty of Economics and Management, University of Sfax. She is a Member of the MIRACL Laboratory. Her research concerns are information systems modelling and business process management (BPM). Currently, her works are directed towards the flexibility of business processes using the version notion.

Fatma Ellouze is currently a contractual assistant in the Department of Computer-science, National Engineering School, University of Sfax, Tunisia. She received her PhD in Computer Science from the Faculty of Economics and Management of the University of Sfax, Tunisia,

in September 2018. She is a Member of the Multimedia, Information systems and Advanced Computing Laboratory, since 2013. Her current research interests include business process management, process modelling, context modelling, ontologies and information systems.

Rafik Bouaziz is Professor Emeritus on computer science at the Faculty of Economic Sciences and Management of Sfax University, Tunisia. He was the president of this University during August 2014 – December 2017, and the director of its doctoral school of economy, management and computer science during December 2011 – July 2014. His PhD has dealt with temporal data management and historical record of data in Information Systems. The subject of his accreditation to supervise research was "A contribution for the control of versioning of data and schema in advanced information systems". Currently, his main research topics of interest are temporal databases, real-time databases, information systems engineering, ontologies, data warehousing and workflows. Between 1979 and 1986, he was a consulting Engineer in the Organization and Computer Science and a Head of the Department of Computer Science at CEGOS-TUNISIA.

Eric Andonoff is an Associate Professor at the University of Toulouse and a Member of the IRIT Laboratory. His research addresses issues in business process management such as (self-) adaptation of processes, context-aware processes, versioning of processes, process-based crisis management and social processes. He is an author of several papers dealing with these issues and published in conferences and journals. He is also involved in several French and European projects.

1 Introduction

Over the two last decades, there has been an increasing use of business process management systems (BPMS) in companies (Dumas et al., 2005). This is mainly due to the fundamental role processes play in companies as they are the support of the alignment between their information systems (IS) and their business strategies (Rolland, 2010; Alotaibi and Liu, 2017). This evolution is mainly due to the maturity of the Business Process Management (BPM) area, which is a well-established research area, defining concepts, methods and rules to help systems support the whole process life cycle, including not only process modelling and enactment but also process analysis, simulation, supervision, monitoring, and discovering (Weske, 2007; Aalst et al., 2005). As a consequence, many standards have emerged such as Business Process Model and Notation (BPMN) (OMG, 2014), which is considered as the de-facto standard for process modelling, and which is easily understood by BPM practitioners. However, BPM systems fail to become widely adopted by companies as they have still important challenges to address. Process flexibility is still one of these challenges (Reijers, 2006; Reichert and Weber, 2012). Indeed, as the economic environment in which companies are involved is more and more dynamic, competitive and open, companies frequently change their processes in order to meet, as quickly and efficiently as possible, new organisational or customer requirements and new law regulations, or to benefit from new collaboration opportunities (Nurcan, 2008).

Generally, process flexibility is defined as the ability of a process to accommodate changes happening in its operating environment (Nurcan, 2008). In order to feature process flexibility and to evaluate the ability of BPMS and process schema/models to support process flexibility, several taxonomies have been proposed in literature. The more suitable one, which serves as a basis for the related work analysis in this paper, is given in (Reichert and Weber, 2012). This taxonomy highlights two different times when process flexibility can take place: flexibility at design-time, which refers to foreseeable changes that can be taken into account in modelled process schema, and flexibility at run time, which refers to unforeseeable changes occurring during process execution. In addition, this taxonomy identifies four needs of flexibility:

1 variability, for representing a process differently, depending on the context of its execution,

2 adaptation, for handling occasional situations or exceptions which have not been necessarily foreseen in process schema,

3 evolution, for handling changes in processes, which require occasional or permanent modifications in their schema, and finally

4 looseness, for handling processes whose schema are not known a priori and which correspond to non-repeatable, unpredictable, and emergent processes.

These last processes require loose specifications.

To address process flexibility in BPM, several contributions have recommended versioning (Chaâbane et al., 2009; Ekanayake et al., 2011; Zhao and Liu, 2013; Ellouze et al., 2016; Lassoued et al., 2016; Ben said et al., 2018). Versioning consists of the definition of several process schema versions for taking into account significant changes on processes and their components (i.e., activities, data consumed and produced, roles involved in their execution). Versioning is acknowledged as to be tailored to address process evolution, process adaptation and process variability, which are the main process flexibility needs in BPM (Reichert and Weber, 2012). In addition, versioning facilitates the migration of running instances from an old process schema to a new one. In fact, changes performed on

such instances may affect already executed activities, making the migration impossible: then versioning is very useful as it allows the running of several instances of the same process according to different schema.

Particularly, contributions recommending versioning for process flexibility have addressed the modelling of versions in BPM. Some of them have introduced versioning models (Ekanayake et al., 2011) and graphs, e.g., Versioned Preserving directed Graph VPG (Zhao and Liu, 2013). Others have proposed specific meta-models supporting the modelling of versions (Chaâbane et al., 2009; Ellouze et al., 2016; Lassoued et al., 2016; Ben Said et al., 2017). Ben Said et al. (2014, 2015) have proposed BPMN4V (BPMN for Versions) to support process versioning. They have introduced the BPMN4V meta-model, which is an extension of the BPMN2.0 meta-model, supporting the modelling of versions of private processes, version of collaborations and versions of choreographies. They have also taken into account the dynamic aspects of versions as they have defined a set of operations for version management and a set of patterns for making changes easier to perform (Ben Said et al., 2015). This contribution is interesting since the recommended approach for versioning takes into account the three main process dimensions (i.e., the behavioural, informational and organisational dimensions) and considers both intra- and inter-organisational processes. In addition, the BPMN4V-Modeller implementing the BPMN4V meta-model has been implemented as an extension of the BPMN-Modeller.

However, to address process versioning in a comprehensive way, it is not enough to just consider the version modelling issue, it is also necessary to address the reuse/selection of modelled versions. Indeed, if a large number of versions co-exist together, BPM practitioners have to face the problem of selecting/retrieving, among different versions of their processes, activities, data and/or roles, the most appropriate ones to a given situation. This may be the case both at run-time if practitioners have to select the process version to be executed among several process versions or/and at design-time if they have to select a version (e.g., a version of activity, a version of role, a version of data) that has to get involved in another version.

Several languages have been defined to query process schema and these languages could be used to select versions in BPM. These query languages are classified in Wang et al. (2014) as follows:

1 structural languages, in which the user requirement is defined as a graph and the retrieved process models must contain this graph syntactically,

2 behavioural languages, in which the user requirement is defined as a sequence of ordered activities and the retrieved process models must include this sequence of ordered activities and

3 contextual language, which is based on a given descriptive information about a particular process.

However, as stated in Ellouze et al. (2017), these query languages for processes have to be extended for version selection as this latter is mainly based on descriptive information. This descriptive information is modelled in the contextual dimension of processes, which is another important dimension of processes and which gathers the minimum of elements that impact the design and the execution of a process (Rosemann and Recker, 2006).

Thus we argue in this paper that version selection in BPM should be supported by a contextual language, i.e., a language using contextual dimension of processes for version retrieval. To define such a language, we have to deal with two issues: context modelling for versions in BPM and context querying.

Regarding context modelling for versions, our purpose consists of adding contextual dimension to BPMN4V private processes and BPMN4V collaborations. This contextual dimension is intended to specify the context in which the (process, activity, data or resource) versions must be used. As the vocabulary used in this definition may differ from one designer to another and/or from one company to another, we also intend to define an ontology supporting multiple vocabularies for context element specification and to exploit this ontology to enhance context modelling in order to make version retrieval more effective. Context enhancing consists of deducing new contextual information based on semantic relationships between contextual information.

As for context querying, it consists of retrieving and visualising versions from queries expressed by BPM practitioners. These latter do not necessarily have knowledge of query languages used to query processes and versions of processes (e.g., PQL). To make this step easier, we recommend a new language for querying versions based on their context.

To sum up, this paper addresses

1 context modelling for versions of BPMN processes

2 version querying using context.

To do so, we recommend the use of BPMN4V as the starting point of this modelling. The paper extends this contribution as follows. First, it introduces an extension of BPMN4V meta-model, called BPMN4V-Context, and an appropriate ontology, called Context-Process ontology, for version context modelling. Second, it introduces a new language, called BPMN4V-QL, for querying versions based on their context and illustrates the implementation of this language in the BPMN4V-Modeller.

Accordingly, the paper is organised as follows. Section 2 provides a state-of-the-art of contributions addressing process versioning, process context modelling and process querying. Section 3 presents the background of the paper; first it gives a case study from the maritime domain, which will be used to illustrate the paper contributions, and second it presents BPMN4V which is an extension of BPMN for private processes and collaboration

versioning. Section 4 introduces the BPMN4V-Context meta-model supporting context modelling for versions of private processes and of collaborations. Section 5 is dedicated to the presentation of the Context-Process ontology used to enhance context of versions modelled according to the BPMN4V-Context meta-model. Section 6 presents BPMN4V-QL a language to support querying versions based on their context. Section 7 presents the extension of the BPMN4V-Modeller to support the querying of BPMN4V processes using context. Finally, Section 8 concludes the paper and gives some directions for future works.

2 Related works

In this section, we present the most important contributions addressing process versioning, context modelling for processes and process querying. Then we give a comparison and a discussion about the presented works.

2.1 Process versioning

Several works dealing with the version-based approach have addressed the modelling of versions in BPM (e.g., Chaâbane et al., 2009; Ellouze et al., 2016; Lassoued et al., 2016; Ben Said et al.; 2017). They have introduced specific versioning models and/or meta-models for capturing process changes over time. These version-based meta-models support the modelling of versions for concepts relevant to the behavioural dimension of processes only (e.g., Ekanayake et al., 2011, Zhao and Liu, 2013), or also to the informational and organisational dimensions of processes (e.g., Chaâbane et al., 2009; Ellouze et al., 2016; Lassoued et al., 2016; Ben Said et al.; 2017). Moreover, most of these contributions only addressed versioning of processes internal to companies (e.g., Chaâbane et al., 2009; Ekanayake et al., 2011; Zhao and Liu, 2013; Lassoued et al., 2016) while only a few of them (e.g., Ellouze et al., 2016; Ben Said et al., 2017) also addressed versioning of processes crossing the boundaries of companies. For example, the meta-model version business process meta-model (VBP2M) introduced in Ellouze et al. (2016) supports modelling of versions of processes, activities, roles and informational resources for both internal processes and processes crossing the boundaries of companies. However, VBP2M is not a standard meta-model for BPM and its recommended notation is not likely to be used by BPM practitioners. To overcome these drawbacks, the BPMN4V meta-model recommended in Ben Said et al. (2014, 2016) extends the BPMN2.0 standard to integrate version modelling capability; BPMN4V supports the definition of versions of processes, collaborations and choreography.

Regarding version management, some of the works have defined a set of operations enabling the creation, update and derivation of versions (Chaâbane et al., 2009; Zhao and Liu, 2013; Ellouze et al., 2016; Ben Said et al., 2017). However, the version management has been discussed from a theoretical perspective, but practical applications are very limited (Reichert and Weber, 2012; Natschläger et al., 2016). For instance, the BPM tools in Signavio (2015) and Scheer (2015), which support process version modelling, especially lack functionality regarding version management: versions are created separately, without any connection between parent and child version(s) (i.e., no derivation hierarchy). On the other hand, the BPMN4V-Modeller (Ben Said et al., 2018) is an editor for modelling and handling BPMN4V versions. It allows

1 the modelling of versions of processes, collaborations and choreographies according to BPMN4V meta-model

2 the management of the modelled versions using operations (creation, derivation, update, froze).

In this tool, relationships between versions, i.e., derivation hierarchy, are well established. However, BPMN4V-Modeller does not support version querying and must be extended to do so.

2.2 Context modelling

Process context is defined in Rosemann and Recker (2006) as "the minimum set of elements containing all relevant information that impacts the design and execution of a process". To classify these context elements, several taxonomies have been introduced (e.g., Wang et al., 2004; Rosemann et al., 2008; Saidani et al., 2015; Brocke et al., 2016). We outline the largest one, described in Rosemann et al. (2008), which distinguishes four types of context:

1 immediate context, which covers elements on process components, namely context of activities, events, and resources

2 internal context, which includes elements on the internal environment of an organisation that impacts its processes

3 external context, which encompass elements relating to external stakeholders of organisations, and finally

4 environmental context, which contains elements related to external factors.

Other works (e.g., Santos et al., 2010; Chaâbane et al., 2009) only distinguish two types of information to represent process context: functional information related to the process components (activities, events, resources…) and non-functional information related to the process quality (safety, security, cost, time…). In this work, we rather adopt the taxonomy introduced in Rosemann et al. (2008), which is more comprehensive.

Regarding context modelling, several approaches have been introduced. The most relevant ones are the key-value approach, the graphical model approach and the ontology-based approach.

In the key-value approach, a context element is represented as a pair (attribute, value) (Dey et al., 2001). The attribute represents the name of the context element while the value represents the current value of this element. This approach uses the most simple data structure for

modelling contextual information. However, it lacks expressiveness since it does not represent the relationships between contextual information.

The graphical model approach uses models and meta-models to model process context (Sheng and Benatallah, 2005; Saidani and Nurcan, 2009; Nie et al., 2014). Although this approach allows modelling the relationships between context elements, it does not offer a mechanism to express semantic relationships such as synonyms and antonyms. However, when a process crosses the boundaries of a company, a semantic interoperability must be ensured between involved partners and the lack of semantic relationship is a drawback.

This drawback is overcome in the ontology-based approach. Indeed, this approach provides a strong source of semantic reasoning and conflict resolution thanks to its capabilities of formal expressiveness and reasoning techniques (Wang et al., 2004; Saidani et al., 2015; Hoang et al., 2014; Lassoued et al., 2016). According to the survey of Strang and Linnhoff-Popien (2004), the ontology based approach is considered as the most promising approach for the context modelling of processes.

Regarding contributions addressing context modelling in BPM, we mention (Hallerbach et al., 2010; Saidani et al., 2015; Lassoued et al., 2016). First the authors of (Hallerbach et al., 2010) have recommended the Provop approach, which helps define context-aware process configurations. This approach consists of a base process, a set of options and a context model. In Provop, this context model comprises a set of context elements, each of which represents one specific dimension of the process context and is defined by a name and a value range. In the configuration phase, each defined option is associated with a context rule condition which is evaluated based on the data describing the process context. Second, the authors of Saidani et al. (2015) introduced a BPM ontology to facilitate the adaptation of processes to different contexts. This ontology is modelled following two abstraction levels:

1 the first one considers an upper context ontology that captures general concepts about contexts, and

2 the second level considers a number of domain-specific ontologies each of which is obtained by instantiating the upper one according to a given domain.

In addition, the authors proposed inference rules for context reasoning to both check the consistency of ontologies and deduce new inferred contexts. However, elements described in these ontologies did not refer to all types of the Rosemann's taxonomy (e.g., the external context). Third, in Lassoued et al. (2016), the authors have addressed context modelling only for versions of the intra-organisational processes (i.e., internal processes). They proposed a specific process context meta-model that focuses only on immediate and internal context of Rosemann's taxonomy. However, it is important to consider also the other contexts defined by this taxonomy as well as inter-organisational processes. In addition, the main drawback of this contribution is its non-conformity with the BPM standards. That is why we advocate the extension of BPMN4V, which supports versioning of BPMN processes, collaborations and choreography, with the notion of context.

2.3 Process querying

Recommended approaches for querying process models can be classified into three types (Wang et al., 2014):

1 structural querying which corresponds to queries defined as a sub-graph of the process to be selected

2 behavioural querying which is based on a given sequence of ordered activities

3 contextual querying which is based on a given descriptive information (i.e., metadata describing the context) about a particular process.

Note that literature survey demonstrates the lack of, and the need for, dedicated precise contextual-based process querying (Wang et al., 2014). This conclusion is also drawn more recently in Polyvyanyy et al. (2017). Indeed most contributions address structural and behavioural query, and querying using context, which is of major importance for version retrieval, is partially addressed.

The relevant contributions about structural querying are especially discussed in Awad et al. (2008) and Beeri et al. (2008). The former introduced a visual query language, called BPMN-Q, using BPMN notation extended with additional attributes. BPMN-Q queries are first defined as a graph, and then converted into semantically expanded queries using the ontology eTVSM that exploits WordNet. As for the latter, it proposed BP-QL query language, as an extension of XQuery language, for BPEL4WS process definitions. The authors recommend a graph-based query language in the form of a BPEL script, to look for a sub-graph isomorphism in a repository of BPEL scripts.

Behavioural querying focuses on behaviours induced by the activities of the process models. Typical examples are querying techniques based on Petri nets (Polyvyanyy et al., 2014), YAWL nets (Jin et al., 2013), and process query languages, e.g., APQL (Ter Hofstede et al., 2013).

There are only few contributions that support contextual querying of processes. We mention (La Rosa and Dumas, 2008; Lu et al., 2009; Kumar and Yao, 2012). These contributions have addressed querying of process variants. However, these works did not advocate an ontology based approach either for modelling or for querying the context. First, in La Rosa and Dumas (2008), a new query approach, called Questionnaire, is introduced. This is applied to lead the user to configure process variants based on formal conceptualisation of domain knowledge. This approach considers especially the goal of each variant and does not consider other types of Rosemann's taxonomy. Second, in Lu et al. (2009), a query (e.g., find all process variants in which execution duration is less than 3 h, and any performer of senior engineer role was involved) is considered as a collection of one or more contextual information elements. This approach allows retrieving

process variants based on the similarity between the specified contextual information elements and the stored ones. However, the variants cannot be retrieved if the users use in their query synonyms or misnomers to the stored contextual information. Third, Kumar and Yao (2012) propose to describe each variant using a context rule and to base the variant querying on the defined context rules. Therefore, the user specifies SQL queries to query the defined context rules, and the corresponding variant is displayed to the user according to the retrieved rule. In addition, the authors developed specific indexes to faster access to the process models.

Table 1 Evaluation of examined works

Domain / *Criteria* / *works*	*Version-based approach*			*Context modelling*		*Process querying*			
	Derivation hierarchy	*Management of versions*	*Standard meta-model*	*Ontology-based approach*	*Rosemann's Taxonomy*	*Structural*	*Behavioural*	*Contextual*	*Ontology-based approach*
Ekanayake et al. (2011)	+	–	–	–	–		Not addressed		
Zhao and Liu (2013)	+	+	–	–	–		Not addressed		
Ellouze et al. (2016)	+	+	–	–	–	+	+	+	+
Lassoued et al. (2016)	+	–	–	–	+/–	–	–	+	+
Ben Said et al. (2017)	+	+	+	–	–		Not addressed		
Signavio, Scheer	–	+	+	–	–	–	–	+	–
Hallerbach et al. (2010)	–	–	–	–	+/–		Not addressed		
Saidani et al. (2015)	–	–	–	+	+/–	–	–	–	–
Awad et al. (2008) and Beeri et al. (2008)	–	–	–	–	–	+	–	–	+
Jin et al. (2013) and Polyvyanyy et al. (2014)	–	–	–	–	–	–	+	–	–
La Rosa and Dumas (2008), Lu et al. (2009) and Kumar and Yao (2012)	–	+	–	–	–	–	–	+	–

Regarding contextual querying for process versions, we mention (Lassoued et al., 2016). This work has introduced VBPQL, an SQL-like language, for the definition and the manipulation of intra-organisational process version context. This language helps query the context using a domain ontology. The authors justified the use of the ontology to make more efficient the implemented filtering mechanisms, namely the exact comparison, subsumption and plug-in mechanisms. However, this work poorly takes advantage of the ontology as the authors do not implement any reasoning strategy as no rules are modelled. In addition, this work does not illustrate how process designers or users specify queries for contextual querying and how they use the filtering algorithms. Finally, this work does not provide

a specific modeller for modelling and retrieving versions of processes.

Finally, some existing BPM tools ensure contextual querying for process versions. Examples include Aeneis (2014), Signavio (2015) and BPaaS (Scheer, 2015). However, the retrieval mechanisms in these tools are very basic and are confined to the search for versions by keywords, such as process name, publishing date, author's name, revision comment, etc. They fall short of providing any guideline or help to retrieve the process version appropriate for a specified context.

2.4 Comparison and discussion

Table 1 gives an evaluation of the previous contributions with respect to the version-based approach, the context modelling and the process querying support. Regarding the version-based approach, we indicate if the work allows or not

1 the generation of a derivation hierarchy that gives a comprehensive view of the process versions

2 the management of versions through a set of operations

3 the use of a standard notation.

As for the context modelling support, we want

1 to evaluate if the examined contribution advocates or not an ontology-based approach to ensure semantic interoperability

2 to check if the proposed ontology fully or partially covers the context types defined in Rosemann's taxonomy (Rosemann et al., 2008).

Finally, regarding the process querying support, we firstly mention the type of the query (structural, behavioural or contextual) and then we evaluate if the examined contribution advocates or not an ontology-based approach to ensure semantic interoperability through reasoning based on semantic relationships between context elements (e.g., symmetry, transitivity, specialisation). The evaluation is notified as follows: + (–) means that the feature is supported (not supported), while +/– means that the feature is partially supported.

From this table we note that the main drawback of these works lies in the context modelling. If the work of Saidani et al. (2015) succeeded in modelling the context using an ontology-based approach, the proposed ontology does not fully cover the four context types identified in Rosemann et al. (2008), namely the immediate, internal, external and environmental contexts. Moreover, some of these works recommended an ontology-based approach to query the context, such as Awad et al. (2008) and Lassoued et al. (2016), but they did not model the context with an ontology. In addition, a common criticism of these works is that their version-based approach (including the generation of the derivation hierarchy and the management of versions) is not based on a BPM standard, except the work of Ben Said et al. (2017) that recommends BPMN4V to model versions of BPMN private processes and versions of BPMN collaborations. Indeed, BPMN is promoted by the OMG and is more used by the BPM community than specific ad-hoc notations. Thus we propose in this paper to extend BPMN4V (Ben Said et al., 2017) to support the modelling of the context for versions of BPMN private processes and BPMN collaborations and the querying of versions of BPMN private processes and BPMN collaborations using context.

2.5 Aim of the paper

The work presented in this paper aims at overcoming the drawbacks of existing works. First it introduces a conceptual solution based on BPMN2.0 that supports the modelling of the context of intra- and inter-organisational process versions considering all process dimensions and all process components (process, sub-process, activity, data, role). Second it presents a context-based version query language for efficiently retrieving versions using ontology. In fact, in a multi-version environment, each version is suitable for a specific context. Thus it is more efficient to retrieve the suitable version for a given situation defined according to user needs and/or environment requirements, by exploiting their context rather than their structure or behaviour. In addition, the use of ontology is an interesting solution to ensure semantic interoperability especially in an inter-organisational context. Third, we introduce a BPM tool supporting the modelling of version and their corresponding context of use and supporting version retrieval using context.

3 Illustrative case study and background

3.1 Illustrative case study: pipeline installation process

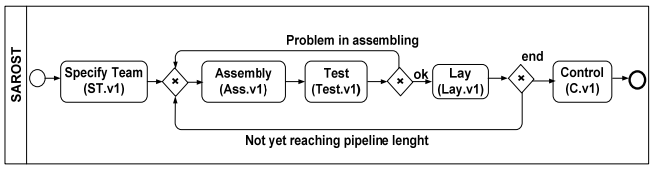
We introduce the subsea pipeline installation (SPI) case study (Ellouze et al., 2017) to illustrate the notions and contributions of the paper. The case study deals with the modelling of a process supporting the installation of subsea pipelines for water or energy transportation. Depending on the context, the process is internal to a single organisation or the result of collaboration between several organisations. Thus it is modelled either as a BPMN private process or a BPMN collaboration. In the following, we present successively versions of the SPI private process and versions of the SPI collaboration

3.1.1 SPI private process

The SPI Private Process (SPI_PP) takes place only within SAROST, a company providing specialised and integrated services in the field of sub aquatic environment. Five activities are considered in this process, namely Specify

Team, Assembly, Test, Lay and Control (cf. Figure 1). First SAROST specifies the necessary team and equipment. Then it proceeds to the assembling and the test of pipes on shore by welders, pipefitters and controllers. The next activity is the laying of pipes offshore by the divers. Finally, when the installation is over, a control has to be performed. Note that the assembling, test and laying have to be repeated until reaching the pipeline length. According to SAROST's domain experts, one version is defined for the Specify Team and Test activities, two versions are defined for the Control activity, seven versions are defined for the Lay activity, and twelve versions are defined for the Assembly activity. As a consequence, twelve versions of the SPI private process are defined. Interested readers can consult (Ellouze et al., 2017) to get additional information on how versions of activities and process are defined.

Figure 1 SPI_PP's first version (SPI_PP.v1)



Each version of the SPI private process depends upon the following context elements:

Sea depth, which can be less than 50 m, between 50 m and 90 m, or longer than 90 m. Indeed, the depth affects both the divers' skills and the diving methods. If the sea depth is less than 50 m, then at least class-2 divers must dive for assembling, while if it is more than 50 m, then at least class-3 divers must dive. Regarding the diving method, diving with umbilical has to be used when depth is less than 50 m, diving with wet bell has to be used when depth is between 50 m and 90 m, and diving with saturation has to be used when depth is more than 90 m.

Pipeline length, which can be less than 10 km, between 10 km and 50 km, or over 50 km. This length affects the number of welders and/or pipefitters involved in assembling.

Installation technique, which can be floating or sliding and that respectively refers to J-Lay and S-lay techniques.

Transported substance, which can be oil, gas or water.

Table 2 gives the context of the different versions of SPI private process. For example, the first version SPI_PP.v1 of SPI private process is defined for the following context: sea depth is less than 50 m, pipeline length is less than 10 km, installation technique is floating and transported substance is water.

3.1.2 SPI collaboration

The SPI Collaboration (SPI) involves three participants: first the Client Company (Client) asking SAROST for the installation of a subsea pipeline, second the SAROST Company and third, the Bureau Veritas (BV) to which the certification of the pipeline can be assigned. Three versions are defined for SPI collaboration. In the first version, SPI.v1 (cf. Figure 2(a)), two participants are involved. The Client initiates the collaboration and sends to SAROST a pipeline installation order. After the pipeline installation, SAROST prepares an acceptance certificate and sends it to the client. SPI.v1 is defined in the following context: *Sea depth* is less than 50 m, *Pipeline length* is less than 10 km, *Installation technique* is sliding and *Transported substance* is water.

Table 2 SPI_PP versions and their contexts

Context version / *Number*	*Sea depth*	*Pipeline length*	*Installation technique*	*Transported substance*
SPI_PP.v1	<50	<10	Floating	Water
SPI_PP.v2	<50	<10	Sliding	Water
SPI_PP.v3	<50	<10	Floating	gas or oil
SPI_PP.v4	<50	10 to 50	Floating	gas or oil
SPI_PP.v5	<50	>50	Floating	gas or oil
SPI_PP.v6	>=50	10 to 50	Sliding	gas or oil
SPI_PP.v7	>90	10 to 50	Sliding	gas or oil
SPI_PP.v8	50 to 90	>50	Sliding	gas or oil
SPI_PP.v9	>90	>50	Sliding	gas or oil
SPI_PP.v10	<50	<10	Sliding	gas or oil
SPI_PP.v11	<50	10 to 50	Sliding	gas or oil
SPI_PP.v12	<50	>50	Sliding	gas or oil

For oil or gas transportation and when the pipeline length does not exceed 50 m, SAROST only proceeds to the installation without doing the Control and the Certification activities. These two activities are subcontracted to the BV. The latter is contacted by SAROST, this is why we have modelled the SAROST send and receive activities: SRC.v1 and RC.v1 (cf. Figure 2(b)).

Finally, in the case of big project involving the installation of long and deep-water pipelines, SAROST only proceeds with the installation. Once the installation is over, it sends back a report to the Client which contacts BV which proceeds itself to the certification of the pipeline (cf. Figure 2(c)).

Table 3 gives the context of the different versions of SPI Collaboration in terms of sea depth, pipeline length, installation technique and transported substance.

Table 3 SPI versions and their contexts

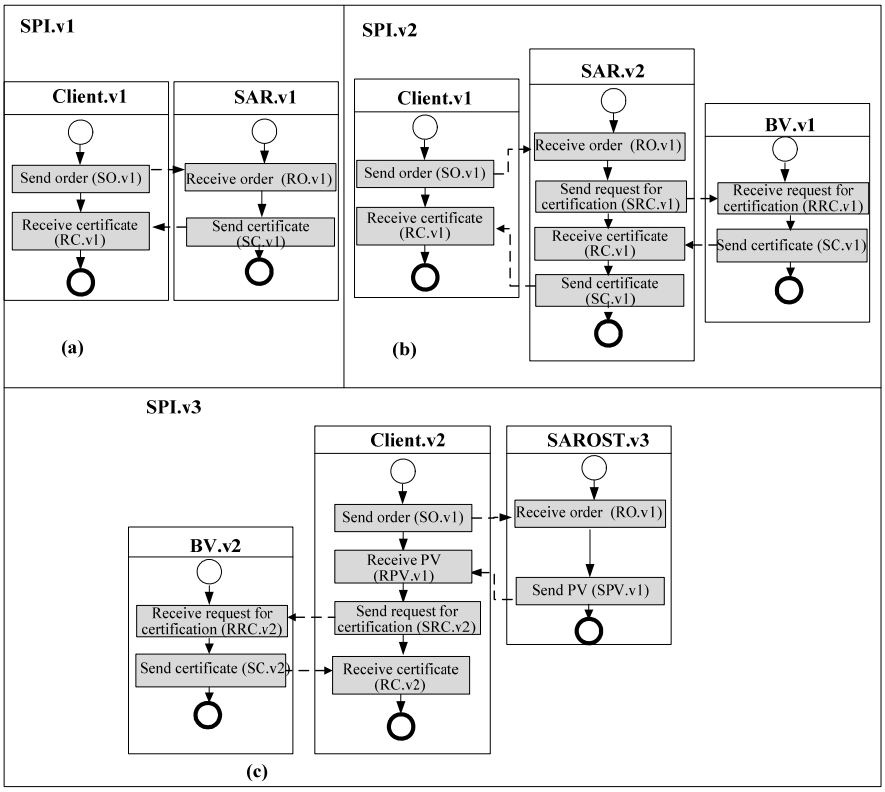
Context version / *Number*	*Sea depth*	*Pipeline length*	*Installation technique*	*Transported substance*
SPI.v1	<50	<10	Floating or Sliding	Water
SPI.v2	<50	<50	Floating	gas or oil
SPI.v3	<50	>50	Floating	gas or oil

3.2 BPMN4V

3.2.1 Static aspect of BPMN4V: the BPMN4V meta-model

The BPMN4V (BPMN for Versions) meta-model (Ben Said et al., 2014, 2016) combines BPMN2.0 concepts for private process and collaboration modelling (OMG, 2014) with the notion of version. In this meta-model, authors recommend handling versions for nine BPMN2.0 concepts in order to model versions of private processes and collaboration: Process, Sub Process, Event, Activity, ItemAwareElement, Resource, ResourceRole, Collaboration and Message. The underlying idea is to keep track of changes occurring to components participating in the description of the way business is carried out. In fact, each of these classes represents a key concept for private processes and collaborations and plays a strong role in their definition. Figure 3 is an UML class diagram of the BPMN4V meta-model: BPMN2.0 concepts are shown in white while concepts related to the versioning are shown in grey.

Figure 2 Three versions of SPI

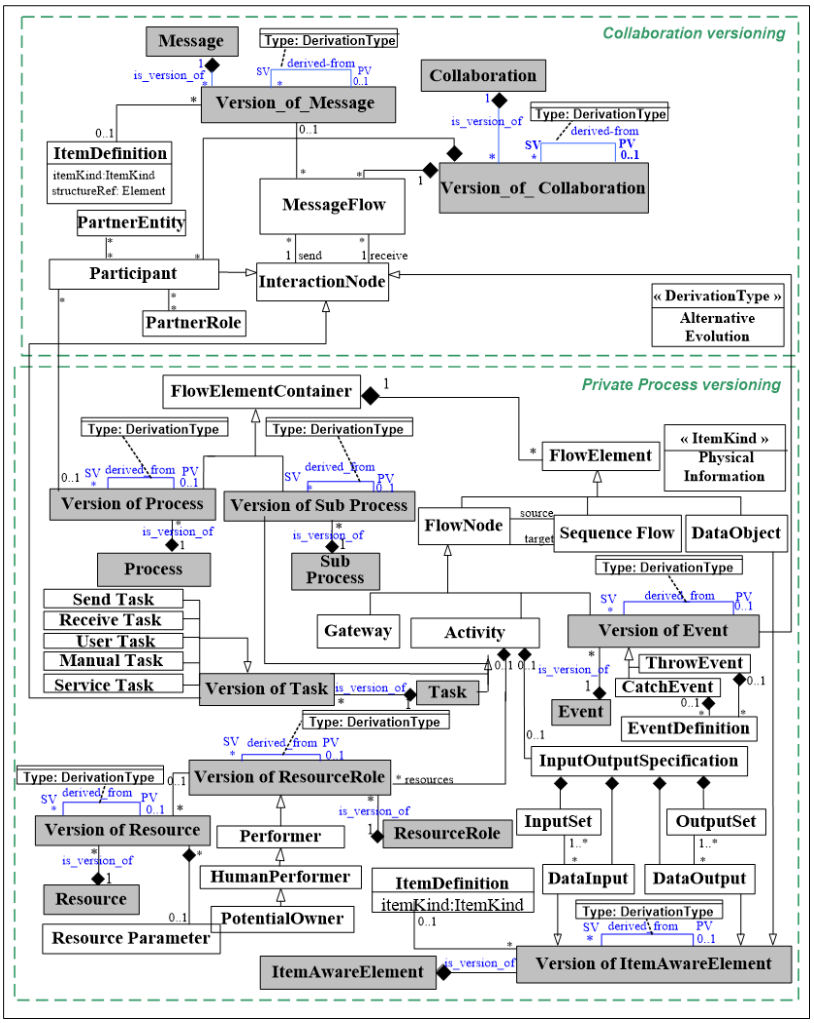


BPMN4V supports the versioning of concepts related to the three main dimensions of private processes, i.e., the behavioural, the informational and the organisational dimensions. The behavioural dimension of a process supports the description of its activities and their coordination along with the events occurring during their 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 occurs 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. It 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). In the frame of the organisational dimension of processes, an activity is performed by a *ResourceRole*, which can refer to a Resource. The latter 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*. The informational dimension of processes is considered in the concept *ItemAwareElement*. This concept references elements used to model the items (physical or information items) that are created, manipulated and used during process execution. An ItemAwareElement can be a *DataObject*, a *DataObjectReference*, a *Property*, a *DataStore*, a *DataInput* or a *DataOutput*. Thus BPMN4V considers the versioning of the following concepts: *Process, Sub Process, Event, Activity, ItemAwareElement, Resource, ResourceRole.*

BPMN4V also supports the versioning of concepts related to collaboration. In a collaboration, each involved partner is seen as a participant that represents a *PartnerEntity* (e.g., a company) or a *PartnerRole* (e.g., a buyer). A participant is often responsible for the execution

of a process. A process involved in a collaboration is a FlowElementContainer that may contain SequenceFlow and FlowNode. Furthermore, within a collaboration, participants are prepared to send and receive *Messages* within *Messageflows*. A messageflow illustrates the flow of messages between two interaction nodes. An *InteractionNode* is used to provide a single element as the source (send relationship) or the target (receive relationship) of a message flow, and therefore of a message. It can be a participant, a task or an event. Therefore, BPMN4V supports the versioning of collaboration and message concepts.

Figure 3 BPMN4V Meta-model for private process and collaboration (see online version for colours)



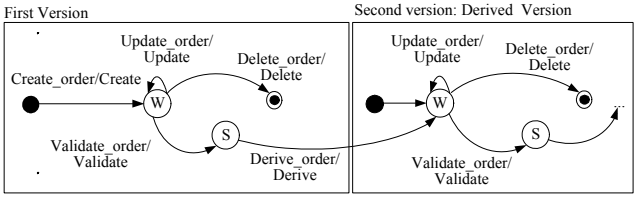
To take into account the notion of version in BPMN2.0, Ben Said et al. (2014) have recommended modelling, for each versionable concept, both the concept itself and the versions it gathers. As suggested in Ben Said et al. (2014) the authors advocate the modelling of the process in a class and its versions in a separate class. Two specific relationships are added between these classes: the *is_version_of* relationship, which makes the link between a concept and its versions, and the *derived_from* relationship, which makes the link between the versions themselves and allows building version derivation hierarchies.

3.2.2 Dynamic aspects of BPMN4V

To handle versions of processes modelled as instances of the BPMN4V meta-model, Ben Said et al. (2018) recommends the use of operations that allow creating, deriving, updating, validating and deleting versions.

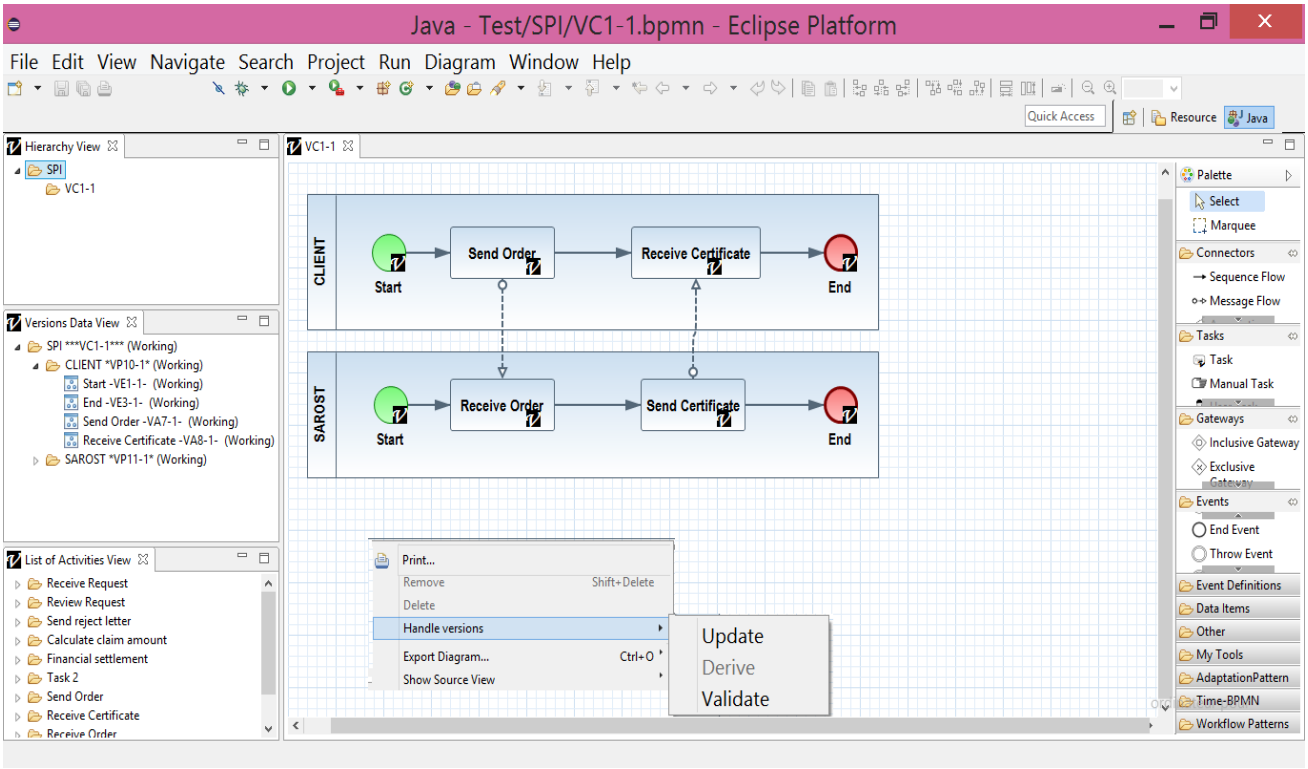
The UML state chart given in Figure 4 indicates when operations for versions are available with respect to the version state. Some of them are available whatever the state of the version on which they are performed, while others are available only in some cases. In this state chart, each operation is described using the notation Event/Operation whose meaning is 'if Event appears then Operation is triggered'.

Figure 4 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. However, 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 for which a new cycle begins.

Figure 5 BPMN4V-Modeller overview (see online version for colours)



3.2.3 BPMN4V modeller

BPMN4V-Modeller is a tool dedicated to the modelling of versions of private processes and collaborations. It allows creating and handling of versions taking into account the recommended extensions of BPMN2.0 and the dynamic aspect of versions. This modeller extends the Eclipse BPMN-Modeller plug-in by integrating new Eclipse views to show version details. Figure 5 gives an overview of BPMN4V Modeller.

The central part of the screenshot (part ❶) is the drawing canvas which provides multiple tabs, each one being used to model and display a separate BPMN diagram (that represents a particular version of private process or collaboration). The right part of the figure (part ❷) represents the Tool Palette, which contains tools that can be dragged onto the drawing canvas to create BPMN elements. The left parts of the figure (parts ❸,❹, and ❺) correspond to the added Eclipse views. More precisely, we have defined three Eclipse views which are Version Data view, List of Activities view and Hierarchy view:

Versions Data view indicates the properties of the active versionable concept: name, version id, version state and version details. For example, the Versions Data view presented in Figure 5 part ❸ indicates that the active version of collaboration, identified by VC1-1, is a working version and corresponds to the first version of the SPI collaboration. In addition, this view details each process, task and event that makes up the considered version. More precisely, this view shows

1 that this version of collaboration is an interaction between the first version of the client process (VP10-1) and the first version SAROST process (VP11-1)

2 that each version of these two processes is composed of a set of versions of tasks and events and

3 that all these versions are in the Working state.

Hierarchy view aims at providing a hierarchical tree oriented view representing the derivation hierarchy of the active versionable concept (e.g., Collaboration or Process) of the drawing canvas. In Figure 5 part ❺, the Hierarchy view shows the derivation hierarchy of the SPI collaboration since the active versionable concept is this collaboration.

List of Activities view, presented in Figure 5 part ❹, displays all the previously modelled tasks and their corresponding versions. This view allows designers to reuse previously modelled versions by dragging and dropping them into the drawing canvas.

In addition to Eclipse views, BPMN4V-Modeller provides the Handle versions contextual menu. This menu holds for each versionable element in collaboration diagram (i.e., Task, Event, Process and Collaboration). For instance, Figure 5 part ❻ shows the contextual menu 'Handle versions' available for versions of collaboration, which implements operations of the state chart (i.e., Update, Derive and Validate).

4 Modelling context of BPMN4V private processes and collaborations

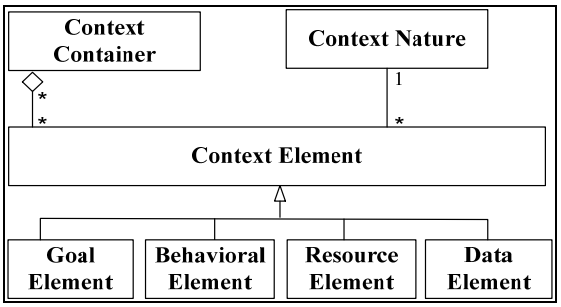
We recommend using the version notion to support process flexibility. Thus several versions can be defined for a process or a collaboration, also for their tasks, sub-processes, events, data exchanged between activities (i.e., ItemAwareElement and Message) and versions for the organisational dimension of processes (i.e., Resource and ResourceRole). Each of the defined versions has to be used in a specific context, i.e., in a specific situation. Therefore, it becomes crucial to consider the contextual dimension of processes to characterise the situation in which these versions have to be used. Indeed, this contextual dimension is fundamental since it serves as a support for version retrieval in BPM. This dimension helps BPM users to retrieve/select the best version when a given situation occurs.

We present below the first contribution of the paper, BPMN4V-Context, which is an extension of BPMN4V considering the contextual dimension of processes. We first introduce a Context meta-model for context description in BPM. Then we present BPMN4V-Context meta-model, which results from the merging of this Context meta-model and BPMN4V meta-model, to model contexts for (process and collaboration) versions. For clarity reasons we separate the description of private process context from the description of collaboration context.

4.1 Context meta-model

The Context meta-model given in Figure 6 allows defining a Context Container as an aggregation of a set of context elements. A Context Element corresponds to a variable characterising a situation, and to which a condition has to be defined. It has a Context Nature, which can be immediate, internal, external or environmental, according to the taxonomy given in (Rosemann et al., 2008). This taxonomy is used to specify the source of each context element. In addition to this taxonomy, we also consider the type of a Context Element, which refers to the dimension to which it belongs to. Thus we consider behavioural Element, i.e., elements related to the behavioural dimension of processes (e.g., activity execution mode, activity duration), Resource Element, i.e., elements related to the organisational dimension of processes (e.g., availability of a resource, experience of a human performer), and Data Element, i.e., elements related to the informational dimension of processes (e.g., data type, data structure). We finally consider Goal Element, i.e., elements describing objectives to be achieved (e.g., objectives of quality, cost, quantity); such type of context elements belong to the intentional dimension of processes (Saidani et al., 2008; Nurcan and Edme, 2005).

Figure 6 Context meta-model



4.2 BPMN4V-context meta-model

4.2.1 BPMN4V-context meta-model for private processes

BPMN4V-Context meta-model results from the merging of the BPMN4V meta-model and the Context meta-model introduced in the previous section. It defines the necessary concepts for modelling contexts of versions of private processes and their versionable components.

Thus in addition to process versions, contexts can be defined for versions of tasks, sub processes, events, resource roles, resources and itemAwareElements. Figure 7 visualises the BPMN4V-Context meta-model showing BPMN2.0 classes in white, versionable classes in grey, context meta-model classes in red and goal and assignment classes in green.

The resulting meta-model links each process to a context container aggregating a set of context elements, corresponding to information from the different dimensions of the considered process: goal elements from the intentional dimension of the process, behaviour elements from the behavioural dimension of the process, resource elements from the organisational dimension of the process, and data elements from the informational dimension of the process. Thus each versionable component of a process can be linked to one or several context elements of its corresponding context container. In addition we define conditions for these elements.

More precisely, goal elements specify objectives of versionable concepts through goal conditions. A Goal Condition is a Boolean expression defining why a version is created. Regarding behavioural, data and resource elements, we specify conditions, called Assignment conditions that allow assigning versionable components. These conditions, described as Boolean expressions, define for each process version the situation in which versions of tasks and events, versions of resources and resource roles and versions of data involved in this process have to be used.

Figure 7 BPMN4V-context meta-model for private process (see online version for colours)

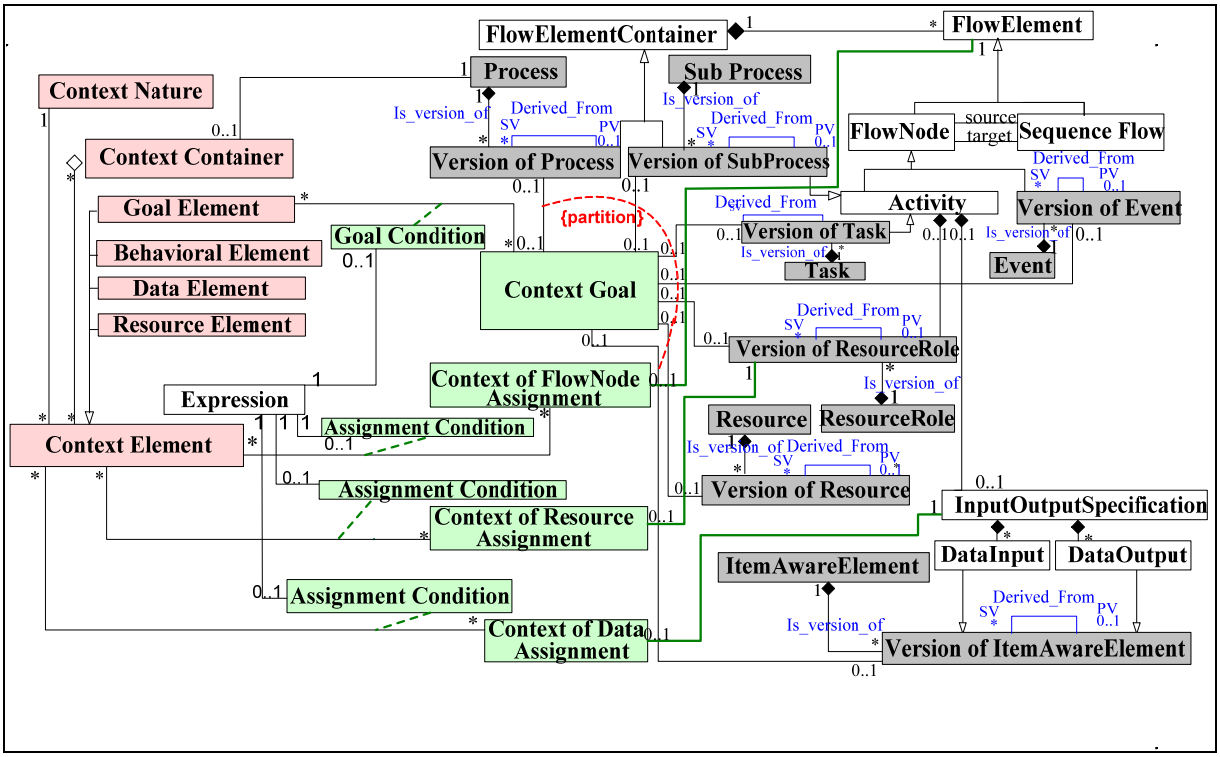
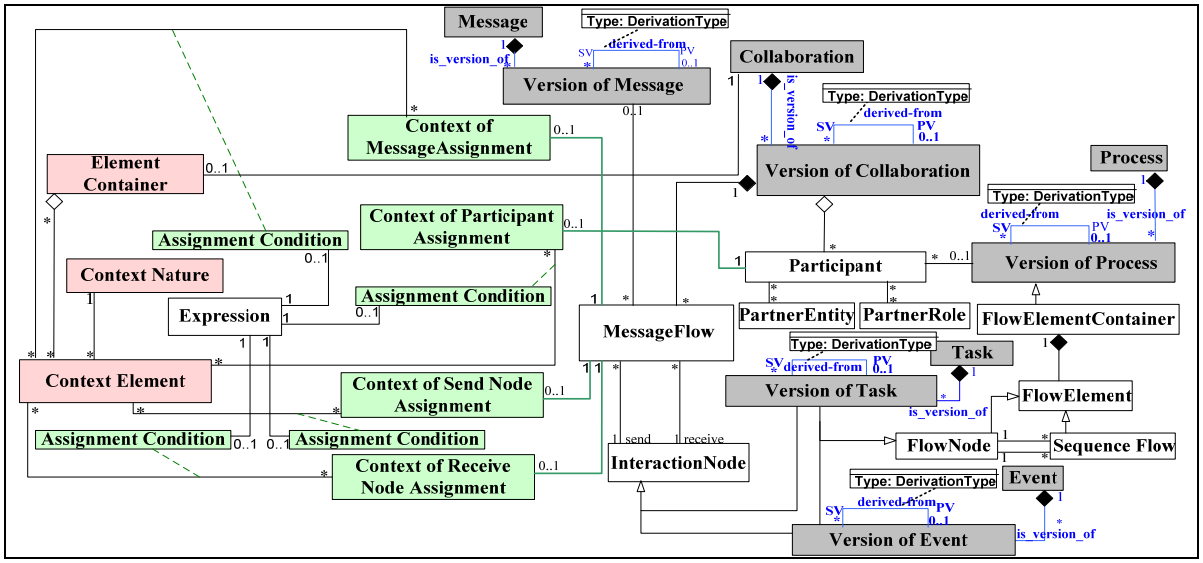


Figure 8 The BPMN4V-context meta-model for versions of collaborations (see online version for colours)



So, Context-BPMN4V meta-model extends BPMN4V meta-model adding four classes used to define assignment conditions within relationships between BPMN components:

- Context Goal, which is related to each versionable classe and thus allows defining the goal of each version.
- Context of FlowNode Assignment, which is related to the class FlowElement and thus allows defining conditions indicating in which situation a version of task (or a version of event) has to be used in a process version or a sub-process version.
- Context of Resource Assignment, which is related to the class Version of ResourceRole and thus allows defining conditions indicating in which situation a version of activity is performed by a version of ResourceRole. Note that Version of ResourceRole is related by a composition relationship to Activity that is a super-class of Version of Task.
- Context of Data Assignment, which is related to the class InputOutputSpecification, and thus allows defining conditions indicating in which situation a version of task consumes or produces versions of ItemAwareElement. Also, InputOutputSpecification

is related by a composition relationship to Activity that is a super-class of Version of Task.

4.2.2 BPMN4V- context meta-model for collaboration

Figure 8 illustrates BPMN4V-Context meta-model for collaboration. This meta-model supports context modelling for versions of collaborations, processes, messages, events and tasks. In this figure, BPMN2.0 classes are shown in white, versionable classes in grey, context meta-model classes in red and assignment classes in green. Note that the collaboration goal is implicitly defined through goals of processes that participate to this collaboration.

BPMN4V-Context meta-model links together Collaboration and Element Container to allow defining the set of context elements that will participate in the definition of the context of each collaboration version. Moreover, this meta-model supports the specification of the context of each versionable component of the collaboration using Assignment conditions. These conditions, described as expressions, define for each version the situation in which it has to be used. Therefore, BPMN4V-Context meta-model specifies four classes for assignment conditions:

Context of Participant Assignment, which is linked to the class Participant and thus allows defining the condition indicating in which situation a participant, representing some process version, is involved in a collaboration version.

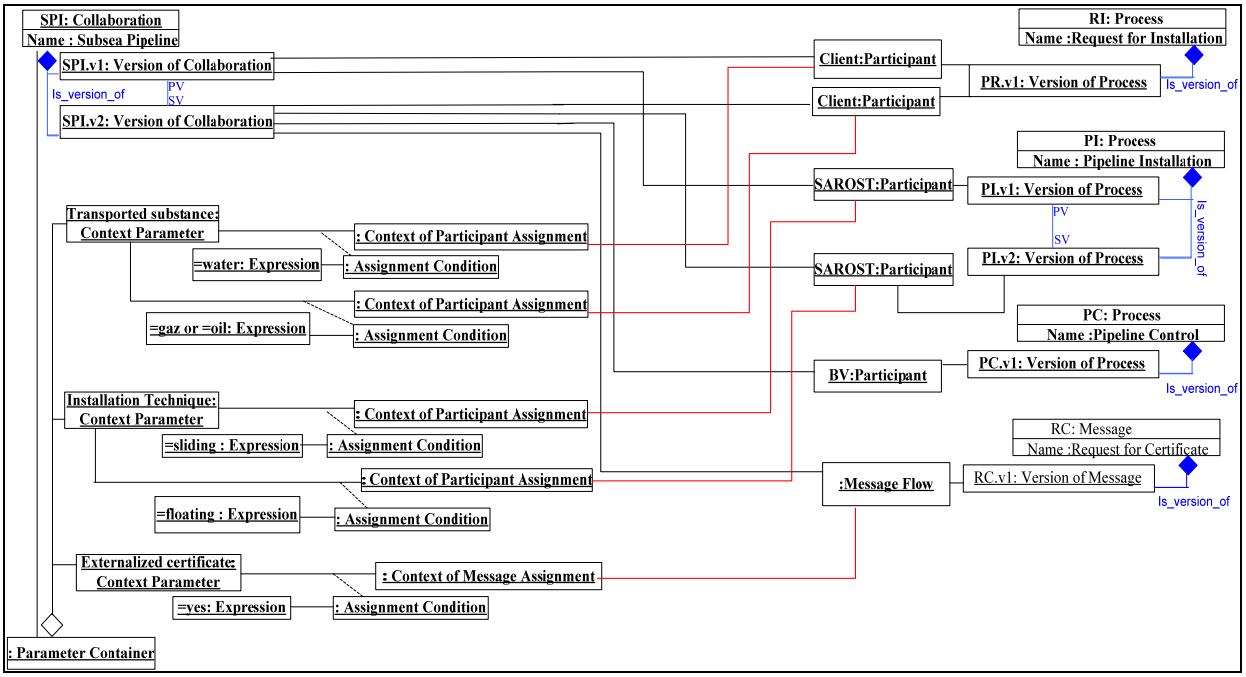
Context of Message Assignment, which is linked to the class Version of Message and thus allows defining the condition indicating in which situation a version of message is sent (or received) within a message flow in a collaboration version.

Context of Send Node Assignment and Context of Receive Node Assignment; which are each linked to the class MessageFlow thus allowing the definition of the condition indicating in which situation an InteractionNode (representing a version of task or a version of event) is sent or received within a message flow in a collaboration version.

4.2.3 BPMN4V-context instantiation

To show how a context model of versions is defined according to the BPMN4V-Context meta-model, we detail in this section an instantiation of this meta-model corresponding to the SPI case study presented in Section 3. More precisely, we illustrate the instantiation of the SPI Collaboration and its different components. Figure 9 partially illustrates this instantiation. It includes two versions of SPI Collaboration (SPI.v1 and SPI.v2), two versions of Pipeline Installation (PI) Process (PI.v1 and PI.v2), one version of Request for Installation (RI) Process (RI.v1), one version of Pipeline Control (PC) Process (PC.v1) and one version of Request for Certificate (RC) Message (RC.v1).

Figure 9 A BPMN4V-context meta-model instantiation (see online version for colours)



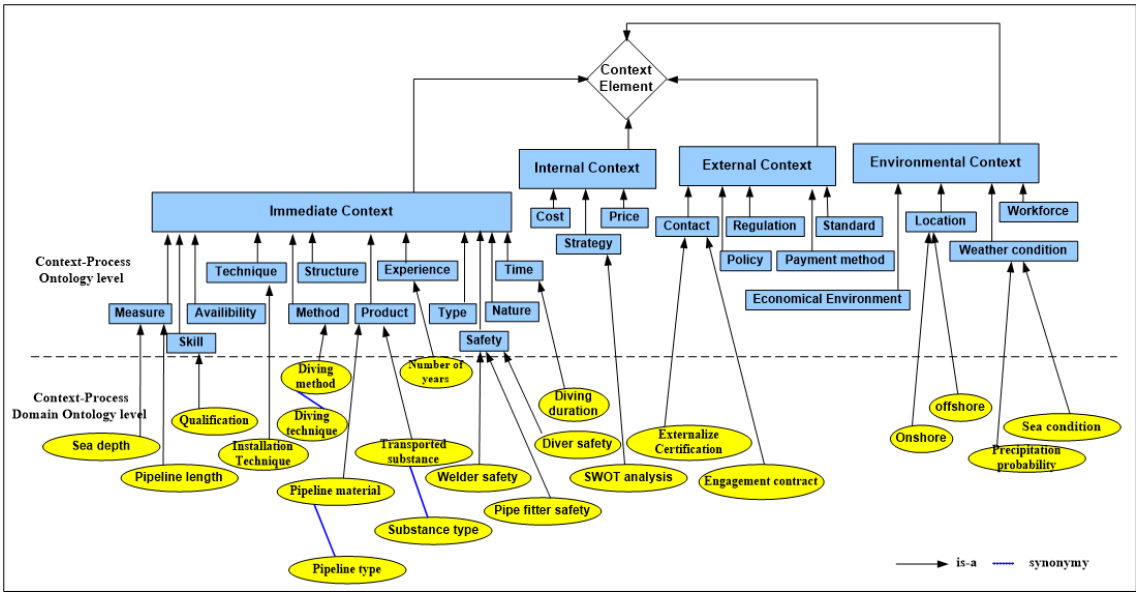
The contexts of SPI versions are defined using the following context elements: sea depth, pipeline length, transported substance, installation technique and externalised certificate. These context elements are instantiations of Element Container related to SPI collaboration. In addition, this figure shows the assignment conditions of participants and messages involved in the versions of SPI Collaboration. For instance, the first version of Pipeline Installation (PI) process holds for the first version of SPI Collaboration in accordance with the context element Installation Technique having as value sliding. Regarding the second version of this process, it holds for the second version of SPI Collaboration in accordance with the context element Installation Technique having as value floating.

5 An ontology to model and enrich the context

5.1 Context-process ontology

Context-process is an upper ontology defining general concepts for context modelling in BPM. It has mainly been designed to address semantic interoperability issues for inter-organisational processes in BPM. This ontology supports types of context recommended in the taxonomy of Rosemann, i.e., immediate, internal, external and environmental contexts (Rosemann et al., 2008). The upper part of Figure 10 gives an excerpt from the context-process ontology where context elements are shown in blue.

Figure 10 The context-process ontology (see online version for colours)



First regarding the immediate context, we define context elements relating to processes and their components, i.e., context elements relating to the behavioural, organisational and informational dimensions of processes. Thus we represent relevant context elements for processes and activities, such as Measure, Safety, Time, Product, and Technique. For instance, Measure includes metrics for quality measurement (e.g., process quality measurement). We also define relevant context elements of

1 informational resources such as Structure and Nature, which specify respectively the structure and the nature of the information (physical e.g., a handbook or information e.g., online book)

2 resources such as Experience and Availability, which specify respectively the experience and the availability of a resource.

Moreover, the internal context covers information on the internal environment of an organisation that impacts its processes. This includes the corporate strategy and related process goal. In addition, the external context captures context elements that are beyond the control sphere of an organisation but still reside within the business network in which this organisation operates. It includes what is related to external stakeholders such as Contract and Payment method. Finally, the environmental context, as the outermost layer, resides beyond the business network in which the organisation is embedded but nevertheless poses a contingency effect on its processes. It includes factors such as Weather condition (e.g., storm season, rain season) and Economical environment.

Context-process ontology is abstract; it does not depend on any domain. To be used, it has to be specialised according to the considered domain. For the Subsea Pipeline domain, which is the domain of the case study presented in Section 3, we have conducted a set of interviews with domain experts to extend the Context-Process ontology with context elements relevant from this domain. The result is the Subsea Pipeline domain ontology presented with yellow colour in Figure 10. The added context elements are defined as a specialisation of abstract context elements of the Context-Process ontology.

In this domain ontology, we provide additional details to the context elements of the upper ontology using specialisation, synonymy and additional relations that can be used for ontological reasoning (e.g., contains relation). For example, the Measure context element is specialised in pipeline length for measuring the length of the pipeline, and sea depth for measuring the depth of the sea in which the pipeline is laid. In addition, we have also defined synonymy for some context elements. For instance, pipeline material and pipeline type are two synonyms for qualifying the material used for the pipeline.

5.2 Enhancing context of versions

BPMN4V-Context meta-model supports specifications of contexts of versions through context elements defined by process designers. However, the jargon used in the definition of these context elements may differ from one group of process users to another and/or from one company to another. This diversity of jargon may cause problems,

especially when the designer defines the context of a particular version using a specific vocabulary and a user tries to retrieve this version using another vocabulary. For instance, if the process designer defines the context of the process version VP1 using the context element 'deep', and a user is looking for VP1's process versions using the context element 'deepness' (which is a synonym of deep), it is not possible to retrieve these versions. Thus we have to ensure semantic interoperability between process designers and process users to make version retrieval more effective.

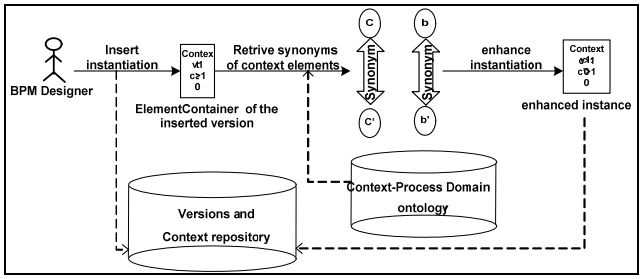
To ensure this semantic interoperability, we propose an enhancing step based on inference mechanisms and allowing the deduction of new contextual information. The idea is to enrich the specific domain context ontology, related to the BPMN4V-Context meta-model, by adding new context elements for versions. This step takes as input the considered domain context ontology, and more precisely the relationships that may exist between the ontology concepts (e.g., semantic relationship), to infer new context elements for versions for which a context has already been modelled as instance of BPMN4V-Context meta-model.

We distinguish two types of context enhancing: a context enhancing after an insertion of new instantiation and a context enhancing after an update of Domain Context ontology. We detail these two types of context enhancing giving algorithms supporting them.

5.2.1 Context enhancing due to insertion of new instantiation

The first type of context enhancing is triggered when the designer stores a new instance of BPMN4V-Context Meta-model in the Versions and Context repository (a repository that contains all previously modelled versions). First we have to interact with the Domain Context ontology to retrieve synonyms of each context element used in this new instantiation. Then we enhance the instantiation by adding new assignment conditions that are defined using the retrieved synonyms. Figure 11 illustrates the first type of context enhancing.

Figure 11 Context enhancing after the insertion of a new instantiation



The algorithm firstTypeofEnhanceInstantiation, presented below, illustrates the first type of context enhancing. This algorithm refers to the Domain Context ontology to enhance the instantiation of a version v having the ElementContainer e. The idea is (1) to retrieve synonyms of each Context Element used in the instantiation of v in the BPMN4V-Context meta-model and (2) update this instantiation by adding new assignment conditions based on the retrieved synonyms. This algorithm uses the following functions:

- getSynonyms(e): returns the list of synonyms of the concept e.
- getAssignmentCondition(e): returns the expression representing the assigment condition related to the Context Element e.
- setAssignmentCondition(ce, ex): affects the expression ex to the Context Element ce.
- getGoalCondition(e): returns the Goal Condition of the Context Element e.
- setGoalCondition(ce, ex): affects the expression ex to the Goal Condition ce.

More precisely, the firstTypeofEnhanceInstantiation algorithm selects for each Context Element elt contained in the ElementContainer e the list of its synonyms stored in the Domain Context ontology. Then it inserts a new instance of the version v accorrding to each retrived synonym.

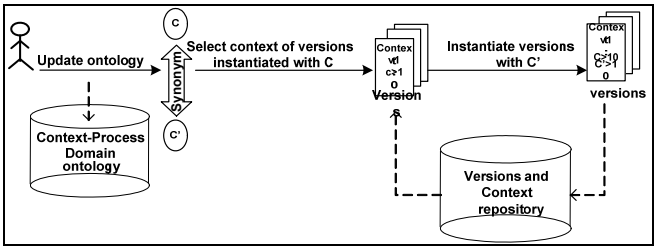
```
firstTypeofEnhanceInstantiation (v: Version,
   e: ElementContainer, onto:
   DomainContextOntology)
ex : Expression
Begin
  For each elt in e
   List L=onto.getSynonyms(elt) /*select all
   synonyms of the context element
                                  elt from
   the ontology*/
       For each l in L
         ex=v.getAssignmentCondition(elt)
   /* select the defined condition of
                                 the context
   element elt for  the version v */
         v.setAssignmentCondition(l,ex) /*
   affect the assignment condition ex
        ex=v.getGoalCondition(elt) /*
   select the defined goal condition of
                                   the
   context element elt for the version v */
         v.setGoalCondition(l,ex) ) /*
   affect the goal condition ex to the

   version v using the context element l */
       End for
  End for
End
```

5.2.2 Context enhancing due to an update of domain context ontology

We also can enhance context of versions when the Domain Context ontology is updated with a new concept C' having a semantic relation with an existing concept C (for instance C' is a synonym of C).Thus we have to interact with Versions and Context repository to select versions whose context is defined using C concept. Then we perform the

instantiations of the selected versions according to the new concept C'. Figure 12 illustrates the second type of context enhancing.

Figure 12 Context enhancing following an update of domain context ontology



The algorithm implementing this type of enhancing is triggered when a new concept related to an existing concept by a relationship (e.g., synonymy) has to be added to the domain context ontology onto. This algorithm uses the following functions:

- addNewConcept(c): adds the new concept c to onto.
- addRelationship (nc, ec, r): adds the relationship r between the new concept nc and the existing concept ec.
- getVersions(ec): returns the list of versions whose context is defined using the contextual element ec.

More precisely, the UpdateOntology algorithm checks if the concept to be added does not exist in the Domain Context ontology. Then it selects all versions that are instantiated using the existing concept and enhances the context of these versions using the new concept.

```
UpdateOntology(existingC: Concept,newC:
   Concept,R: Relationship,onto:
   Relationship)
Begin
 if (newC not in onto) then
   onto.addNewConcept(newC) --add newC to
   the ontologie onto
   onto.addRelationship(newC, existingC, R)
   /* add a new relationship between
                              the added new
   concept and the existing concept*/
   List<Version> lv=getVersions(existingC)
   /* select all versions which
                              context is
   defined using the existing concept */
     For each v in lv
         secondTypeofEnhanceInstantiation(v,
   existingC, newC) /* define a new

   instantiation of v using the new concept
   */
     end for
 end if
End
```

The secondTypeofEnhanceInstantiation algorithm, used in the UpdateOntology algorithm, aims at inserting a new instantiation of the version v defined with the new concept newC in the Versions and Context repository. The idea is to find out the assignment conditions previously defined using the existing concept existingC and then to define new assignment conditions using newC. In addition to the functions getAssignmentCondition(e), setAssignmentCondition(ce, ex), setGoalCondition() and getGoalCondition() previously presented, this algorithm uses the following functions:

- convertToContextElement(c): converts the ontology concept c to a context element of BPMN4V-Context meta-model.
- updateElementContainer(v, e): inserts the context element e in the ElementContainer of the version v.

```
secondTypeofEnhanceInstantiation(v:
   Version,newC: Concept,existingC: Concept)
e1, e2 :ContextElement
ex : Expression
Begin
   e1=convertToContextElement(existingC)
   e2=convertToContextElement(newC)
   updateElementContainer(v,e2) /*adds the
   context element e2 to the element

   container of the version v */
   ex=v.getAssignmentCondition(e1)
   v.setAssignmentCondition(e2,ex)
   ex=v.getGoalCondition(e1)
   v.setGoalCondition(e2,ex)et là
End
```

6 Context based version querying

This section presents the BPMN4V Query Language (BPMN4VQL) we propose for retrieving versions based on context information. First it details the grammar of BPMN4VQL. Then it introduces three different matching types for retrieving versions.

6.1 The BPMN4V query language

We present in the Appendix the BPMN4VQL grammar specified using a context-free grammar expressed using extended Backus-Naur Form. BPMN4VQL is inspired from PQL (Ter Hofstede et al., 2013). PQL and BPMN4VQL serve the same overarching purpose, which is information retrieval. PQL is intended to select process models based on their functional dimension, i.e., tasks that compose these models. For instance, it can fetch the list of processes searching for a particular task. As for BPMN4VQL, it allows the selection of versions based on contextual information. Thus in a BPMN4VQL query the user has to specify a particular situation of context through a set of propositions (or conditions) and the query returns the versions that verify these conditions.

A BPMN4VQL query is composed of three main parts: the SELECT part, the WHERE and the MATCHING part. We give bellow an example of BPMN4VQL query allowing

the selection of versions of processes in accordance with BPMN4VQL grammar.

Select Version_of_Process(*)

Where (Sea_depth **subsume** 50 **Medium**;

Pipeline_length **In** [10;15] **High**;

Installation_technique **equals** “floating” **Low;**

Transported_substance **equals** “gas” **High;**

CONTAINS Version_of_Task Control **WHEN** (controlType equals “deep control”))

Matching Subset ;

The SELECT part specifies the name of the versionable concept to return (e.g., Version_of_Process, Version_of_Task…) and the name of the concept to return (e.g., SAROST process, Control task…). The user can retrieve the versions, whatever the concept name, using the universal (*).

The WHERE part specifies the predicate defining the selection conditions. This predicate can be one proposition, a set of propositions separated with semicolon (;), a component proposition, a set of component propositions or a logical test verifying if a proposition is evaluated to true or false. The definition of a proposition or a component proposition is specially based on context elements. A proposition can be a setPredicate, which is a Boolean expression that has to be evaluated (e.g., WHERE depth Subsume 50 or WHERE depth In [10;50]). A setPredicate is composed of the following three parts: contextElement, situation and pertinenceDegree. The contextElement part is a string that represents the context element name. The situation part is composed of an operator that can be Equal, Subsume or In, and a value which can be a string, an integer or a number. The pertinenceDegree part indicates if the condition defined in the setpredicate part has to be considered with a high, medium or low pertinence degree. This latter serves to sort the query result (i.e., the selected versions) from the most to the less relevant. In addition to setpredicate, a proposition can also be the negation of another proposition (e.g., WHERE NOT (depth equal 20)). Regarding component proposition, it allows the definition of condition related to version components. It is composed of the following parts: CONTAINS versionableConceptName ConceptName WHEN setPredicate. For example, the following component proposition CONTAINS Version_of_Resource Controle_ Agent WHEN (availability equals True) allows the definition of a condition related to availability of the version of Resource Control_Agent.

Finally, the last part of a BPMN4VQL query specifies the matching type that can be Exact, Subset or Superset. An Exact matching type returns versions for which context is defined with exactly the same context elements and conditions of the where part of the query. With a Subset matching type the returned versions are those for which the query context (i.e., conditions of the where part) represents a subset of the version context. The Superset matching type allows the selection of appropriate versions; for each of these versions the set of query context elements and conditions must be included in the sets of the version context elements and conditions. In the next subsections we detail these matching types and introduce algorithms implementing them.

6.2 Matching types

In this section we detail the Exact, Subset and Superset matching types, mainly providing the recommended algorithms for each type.

6.2.1 Exact matching type

Exact matching allows the selection of all versions of a versionable concept having the same context than the one specified in the BPMN4V-QL query. More precisely, exact matching on a query q is defined as follows:

$ExactMatching \equiv \{v \in \{versions\} \mid context_v = context_q\}$ $context_v = context_q$

$=>$i) $\forall ccv=(ContextElement,Condition) \in Context_v$ Then $\exists ccq=(ContextElement, Condition) \in Context_q \mid ccv.ContextElement=ccq.ContextElement$ ^

Verify (ccq.condition,ccv.condition)

and ii) $!\exists\ ccq(ContextElement, Condition) \in Context_q \mid$

$ccq.ContextElement \notin \{Context_v.ContextElement\}$

For instance, let us consider a query q that selects versions of a given process that correspond to the situation defined using the following context {(A, c1), (B, c2), (C, c3)}. A, B and C are context elements and c1, c2 and c3 are their corresponding conditions. An exact matching on q must return the versions of the considered process defined by exactly the same context elements A, B and C for which c1, c2, and c3 conditions are verified. Note that we do not consider the pertinence degree in exact matching type since we suppose that all context elements have the same importance.

Our recommended algorithm, namely ExactMatching, implementing the Exact matching type, uses the following set of functions supporting the comparison of both versions and query context elements and conditions.

- getAllVersions(vc): returns all the versions of the versionable concept vc.
- getContext(): returns the context of a particuler version, represented as a set of couples context element (e.g., Sea_Depth) and condition (e.g., <50).
- getContextElementNumber(): returns the number of context elements used in the definition of a particular context.
- getContextElement(): retunrns the context element used in the definition of the context of a version or a query.

- getCondition(): returns the condition associated to a context element used in the definition of the context of a version (e.g., <50).
- getSituation(): returns the situation defined in a query and associated to a context element (e.g., subsume 20).
- verify(condition): returns true if a situation defined in the query verify the condition specified in the context of a version, otherwise returns false. For example, verify returns true if the situation is 'subsume 20' and the condition is "<50".
- gotoNextVersion(): points to the next version.
- gotoNextQueryContextElement(): points to the next query context element.
- add(v): adds the version v to the selected versions.

```
ExactMatching(queryContext[]:contextConditio
   ns,c:versionableConcept):Versions[]
Begin
Local candidateVersions[],
   selectedVersions [] :Version
      versionContext [] :contextConditions
      find :Boolean
 candidateVersions = getAllVersions(c)
 For each v in candidateVersions
   /*select the context of the candidate
   version v which is composed of

   context elements and conditions*/
   versionContext = v.getContext()
   /* check if the query and the version
   have the same number of context

   elements*/
   if
   (versionContext.getContextElementNumber()
   != queryContext.getContextElementNumber()
   )
        gotoNextVersion()
   end if
   /* compare ContextElements and conditions
   of v and the query*/
   For each cq in queryContext
        find=false
        For each vc in versionContext
              if (cq.getContextElement()=
   vc.getContextElement() and

        cq.getSituation().verify(vc.getCondi
   tion())
               find=true

        gotoNextQueryContextElement()
              end if
        End For
        if (not find)
              gotoNextVersion()
        end if
   end For
   if (find)
        selectedVersions.add(v)
   end if
 End For
 Return selectedVersions
End
```

6.2.2 Subset matching type

Subset matching allows the selection of all versions of a versionable concept whose context is lower than the one specified in the BPMN4V-QL query. More precisely, subset matching on a query q is defined as follows:

$$\text{SubsetMatching} \equiv \{v \in \{versions\} \mid context_v \leq context_q\}$$

$$context_v \leq context_q$$

$$\Rightarrow \text{i)}\ \exists ccv=(ContextElement, Condition) \in Context_v \ \wedge$$

$$\exists\ ccq=(ContextElement, Condition) \in Context_q \mid$$

$$ccv.ContextElement=ccq.ContextElement \wedge$$

$$Verify\ (ccq.condition,ccv.condition)$$

and ii) $!\exists\ ccv(ContextElement, Condition) \in Context_v \mid ccv.ContextElement \notin \{Context_q.ContextElement\}$

For instance, let us consider a query q that selects versions of a given process that correspond to the situation defined using the following context {(A, c1), (B, c2), (C, c3)}. A, B and C are context elements and c1, c2 and c3 are their corresponding conditions. A subset matching on q must return the versions of the considered process whose context is defined using the following sets of context elements and conditions: {(A, c1)}, {(B, c2)}, {(C, c3)}, {(A, c1), (B, c2)}, {(A, c1), (C, c3)}, {(B, c2), (C, c3)}and {(A, c1), (B, c2), (C, c3)}.

For this matching type we have to calculate the weight of the selected version depending on pertinence degrees defined in the query. This weight allows sorting the selected versions from the most relevant to the less relevant. We give below the SubsetMatching algorithm implementing the Subset matching type using the following functions, in addition to the ones previously defined:

- gotoNextVersionContextElement(): breaks loop and points to the next version context element.
- getRelevance(): returns the relevance degree of a particular context element used in the query. Particularly, it returns 1 if the relevance degree is equal to 'High', 0.5 if it is equal to 'Medium' and 0 if the relevance degree is equal to 'Low'.
- add(v, weight): adds the version v to the selected versions. This version will be inserted in the result according to its weight.

```
subsetMatching
   (queryContext[]:contextConditions,c:versi
   onableConcept):Versions[]
Begin
Local candidateVersions[],
   selectedVersions [] :Version
      versionContext [] :contextConditions
      find :Boolean
      weight,n1,n2 :Number
  candidateVersions=getAllVersions(c)
  For each v in candidateVersions
```

```
   /*select the context of the candidate
   version v which is composed of

   context elements and conditions*/
   versionContext =v.getContext()
   n1=
   versionContext.getContextElementNumber()
   n2=
   queryContext.getContextElementNumber()
   /*check if v has more context elements
   then the query*/
   if (n1> n2)
        gotoNextVersion()
   end if

/* compare ContextElements and conditions of
   the candidate version and the

   query*/
   weight=0
   For each vc in versionContext
        find=false
        For each cq in queryContext
           if (cq.getContextElement()=
   vc.getContextElement() and

        cq.getSituation().verify(vc.getCondi
   tion())
                          find=true
                          weight=
   weight+cq.getRelevance()

   gotoNextVersionContextElement()
           end if
        end For
        if (not find)
          gotoNextVersion()
        end if
    End For
     if (find)
        selectedVersions.add(v, weight)
     end if
  end For
  Return selectedVersions
End
```

6.2.3 *Superset matching algorithm*

Superset matching allows the selection of all versions of a versionable concept whose context is greater than the one specified in the BPMN4V-QL query. More precisely, superset matching on a query q is defined as follows:

SupersetMatching≡ {v ∈ {versions} | $context_v \geq context_q$ }

$context_v \geq context_q$

=>∀ ccq=(ContextElement, Condition) ∈ Contextq then

∃ccv=(ContextElement, Condition) ∈ $Context_v$ |

ccv.ContextElement=ccq.ContextElement ^

Verify (ccq.condition,ccv.condition)

For instance, let us consider a query q that selects versions of a given process that correspond to the situation defined using the following context {(A, c1), (B, c2), (C, c3)}. A, B and C are context elements and c1, c2 and c3 are their corresponding conditions.

A superset matching on q must return the versions of the considered process whose context is defined using sets that contains more than A, B, C context elements and their conditions. For example versions with the following sets of context elements and conditions will be returned: {(A, c1), (B, c2), (C, c3), (E, c4)} and {(A, c1), (B, c2), (C, c3), (E, c4), (F, c5)}.

As in the Subset matching type, the Superset matching type calculates the weight of the selected versions to produce a sorted result of the selected versions. We give below the SupersetMatching algorithm implementing the Superset matching type.

```
supersetMatching
    (queryContext[]:contextConditions,c:versi
    onableConcept):Versions[]
Begin
Local condidateVersions[],
    selectedVersions [] :Version
       versionContext []  :ContextElement
       find :Boolean
       weight, n1,n2 :Number

  condidateVersions=getAllVersions(c)
  For each v in candidateVersions

         /*select the context of the
    candidate version v which is composed of

    context elements and conditions*/

    versionContextElements =v.getContextEleme
    nts()
       versionConditions =v.getConditions()
       n1=
    versionContext.getContextElementNumber()
       n2=
    queryContext.getContextElementNumber()
        /*check if the query has more
    context elements then v*/
       if (n1<n2)
         gotoNextVersion()
       end if

        /*compare ContextElements and
    conditions of v and the query*/
        weight=0
        For each cq in queryContext
          find=false
          For each vc in versionContext
                  if (cq.getContextElement()=
    vc.getContextElement() and

    cq.getSituation().verify(vc.getCondition(
    ))
                              find=true
                              weight=
    weight+cq.getRelevance()

    gotoNextQueryContextElement()
                  end if
          End For
          if ( not find)
              gotoNextVersion();
          end if
```

```
        End For
        if (find)
              selectedVersions.add(v)
        end if
  End For
  Return selectedVersions
End
```

7 BPMN4V-Modeller for context modelling and querying

In this section, we introduce the extensions we have implemented to the BPMN4V-Modeller for considering versions context modelling and querying. We remind that BPMN4V-Modeller is a dedicated tool for modelling and handling versions. We detail in the following the new functions of this editor for modelling context of versions and querying versions based on their context.

7.1 Context modelling using BPMN4V-Modeller

Regarding context modelling, we recommend a new Eclipse property tab, named "Context Information", defined for each versionable concept. Figure 13 gives a screenshot of the BPMN4V-Modeller that illustrates the context information property table related to the second version of the SPI collaboration (VC1-2). The left part of this table visualises the context of the selected version. It shows the list of context elements and conditions that define the context of this version. Particularly, this list contains the name, value, nature (i.e., immediate, internal, external or environmental context) and type (i.e., Goal, Behavioural, Data or Resource) of each context element used in the definition of the context of the selected version. Buttons above allow manipulating context elements by adding, deleting, moving or editing them. In the right part of this view, the user can either introduce a new context element and condition by defining its name, value, type and nature or edit an existing one.

Once the context is defined in the Context Information tab, the user can save it using the save button. Thus this context is inserted in the Versions and Context repository and the first type of context enhancing is applied (cf., Section V.1.2).

7.2 Context querying using BPMN4V-Modeller

Context querying BPMN4V-Modeller provides two different Graphical User Interfaces (GUI) for specifying and executing BPMN4VQL queries that are available through the Query context menu. The first one, shown in Figure14, is intended to be used by practitioners that are not necessary computer scientists. It allows the definition of non-complex queries with a limited number of propositions. In this GUI the user has to

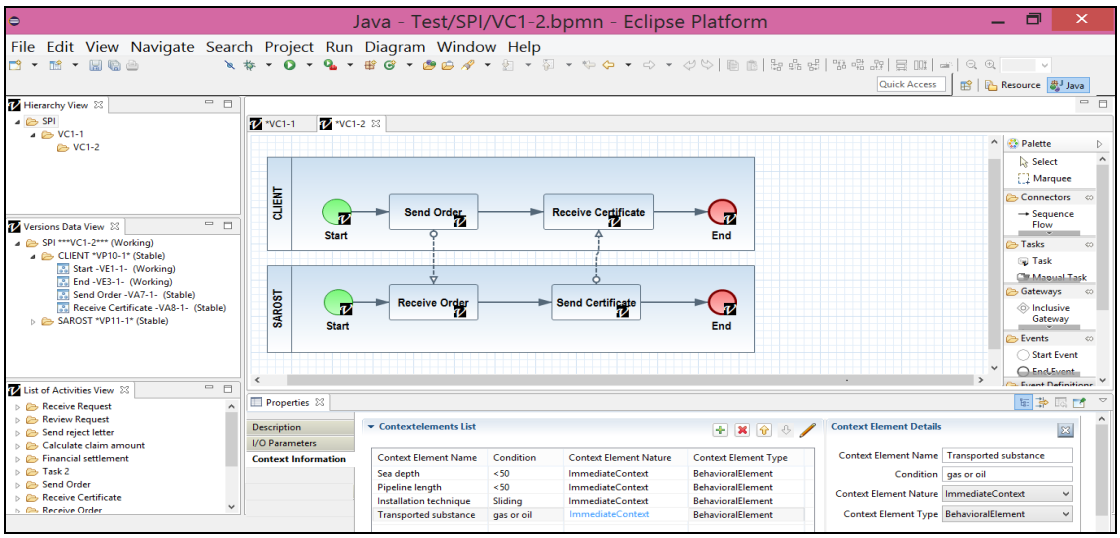
1 specify the versionable concept to retrieve

2 choose the corresponding matching type

3 provide propositions defining the current situation to select.

When clicking the Execute button, a BPMN4VQL query is generated and executed. As indicated before, for the matching types Subset and Superset, the result of the query is sorted by relevance degree, as shown in the result part of the GUI of Figure 14. The user can also visualise the selected versions by clicking on the Visualise version button.

Figure 13 Modelling context with BPMN4V-Modeller (see online version for colours)



The second type of GUI illustrated in Figure 15 allows the definition and the execution of queries with respect to BPMN4VQL grammar. The user has first to introduce his query in the Query part and then click the Execute Query button. Then, a parsing step is performed to check if the query is syntactically correct, and if so, it is executed and

the result is shown in the Result part. Again the query result is sorted according to the relevance degree of the used context elements when the specified matching type is Subset or Superset. Note that the user can see a graphical representation of a resulting version by selecting the version and clicking on visualise button.

Figure 14 GUI for querying versions (see online version for colours)

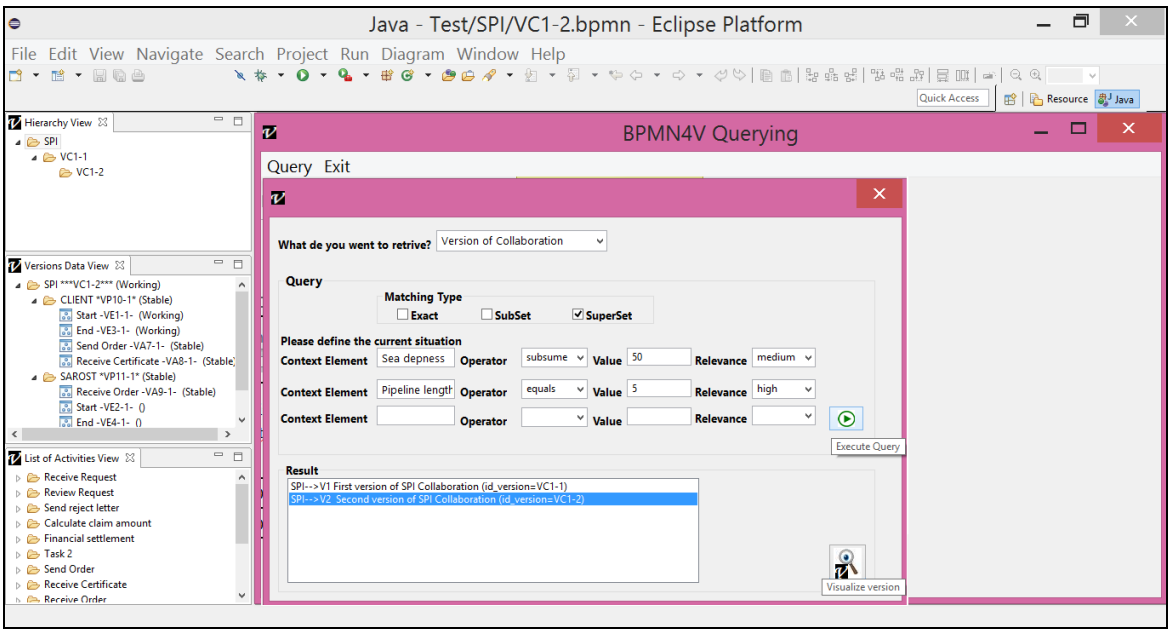
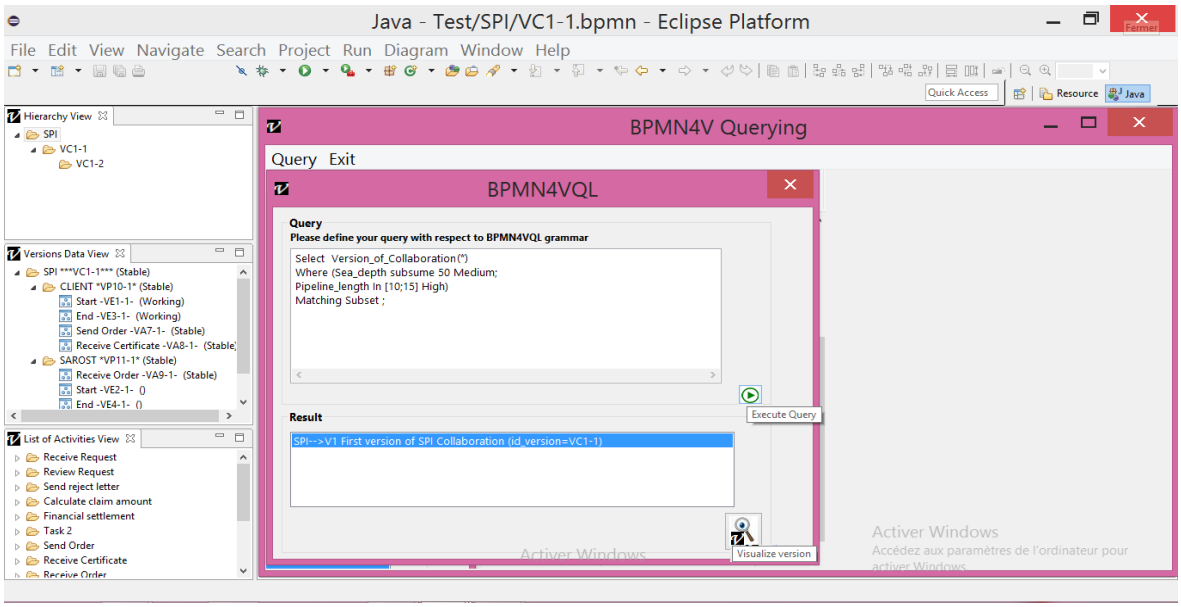


Figure 15 GUI for specifying and executing BPMN4VQL queries (see online version for colours)



8 Conclusion

This paper has addressed the modelling and the querying of context of versions of BPMN processes. Actually, in a multi-version environment, in which a large number of versions can co-exist, BPM practitioners have to face the problem of retrieving, among different versions, the most appropriate one to a given situation. In this paper, we claim that the context notion is very helpful for version(s) retrieval as it helps describe the situation in which versions have to be used. In fact, the problem of retrieving version may arise both at run-time, if users have to choose the versions of the considered process or collaboration to be executed, and at design-time, if BPM designers have to select a particular element version (e.g., a version of task) to participate in the modelling of another version (e.g., version of process).

To address this issue, we recommend a context based approach that provides support for (1) modelling of both versions of BPMN private processes and BPMN collaborations and their corresponding contexts, and (2) version context-based querying. Regarding context

modelling, this paper recommends on the one hand (1) the BPMN4V-Context meta-model, which is an extension of the BPMN meta model considering versions of private processes and collaborations and their contexts, and (2) the Context-Process ontology, which is an upper ontology introducing abstract concepts for context modelling in BPM. On the other hand it recommends an enhancing step that aims to enrich BPMN4V-Context meta-model instantiation using Context-Process ontology. Regarding context querying, this paper proposes BPMN4VQL, a dedicated language for querying versions based on their context. Finally, the paper reports on the implementation of both the meta-model and the language in the BPMN4V-Modeller, which is an extension of the already existing, eclipse plug-in BPMN-Modeller. BPMN4V-Modeller allows modelling and handling versions and their context according to BPMN4V-Context meta-model and querying versions using BPMN4V-QL.

The key strengths of our contribution are the following. First, the recommended solution is based on a standard, the BPMN standard, which is considered as the de facto standard for process modelling, and thus should be more easily used. Second, versions querying is described using BPMN4VQL language that facilitates finding out the appropriate version. This language allows BPM users to find versions based on their context instead of their structure. Thus versions have high chance to be retrieved thanks to the situation defined by the user based on contextual information. Finally, BPMN4V-Modeller proves the feasibility of the proposed solutions since it implements the recommended BPMN2.0 extensions and provides GUI for querying versions. This editor facilitates modelling and handling of versions of processes and collaborations and allows to interpret BPMN4V-QL expressions in order to efficiently retrieve versions based on their context.

Regarding future works, we have planned to revisit our versioning model to give more semantic to the derivation relationship, represented as a derivation hierarchy, which links the versions together. In our model, the derivation relationship, and thus the derivation hierarchy, has no semantic. However, when deriving (i.e., creating) a version from another one, it is possible, considering the context of the derived version, to indicate whether this version is an evolution or an alternative (i.e., a variant) of the version from which it is derived. By doing this, we will have fully exploited the notion of context for versioning in BPM.

References

Aalst, W., Weske, M. and Grünbaur, G. (2005) 'Case handling: a new paradigm for business process support', *Data Knowledge Engineering*, Vol. 53, No. 2, pp.129–162.

Aenies (2014) https://www.intellior.ag/wp-content/uploads/Aeneis-Folder_EN.pdf

Alotaibi, Y. and Liu, F. (2017) 'Survey of business process management: challenges and solutions', *International Journal of Enterprise Information Systems*, Vol. 11, No. 8, pp.1119–1153.

Awad, A., Polyvyanyy, A. and Weske, M. (2008) 'Semantic querying of business process models', *Proceedings of the 12th International Conference on Enterprise Distributed Object Computing*, München, Germany, September 2014, pp.85–94.

Beeri, C., Eyal, A., Kamenkovich, S. and Milo, T. (2008) 'Querying business processes with BP-QL', *Journal of Information Systems*, Vol. 33, No. 6, pp.477–507.

Ben Said, I., Chaâbane, M.A., Bouaziz, R. and Andonoff, E. (2016) 'A version-based approach to address flexibility of BPMN collaborations and choreographies', *Proceedings of the 16th International Conference on e-Business,* Lisbon, Portugal, pp.31–42.

Ben Said, I., Chaâbane, M.A., Bouaziz, R. and Andonoff, E. (2017) 'BPMN4V for modelling and handling versions of BPMN collaborations and choreographies', *Communications in Computer and Information Science*, Vol. 764, pp.1–25.

Ben Said, I., Chaâbane, M.A., Andonoff, E. and Bouaziz, R. (2018) 'BPMN4VC-modeller: easy-handling of versions of collaborative processes using adaptation patterns', *International Journal of Information Systems and Change Management*, Vol. 10, No. 2, pp.140–189.

Ben Said, I., Chaabane, M.A., Andonoff, E. and Bouaziz, R. (2014) 'Extending BPMN2.0 meta-models for process version modelling', *International Conference on Enterprise Information Systems*, April, Lisbon, Portugal, pp.384–393.

Ben Said, I., Chaabane, M.A., Bouaziz, R. and Andonoff, E. (2015) 'Flexibility of collaborative processes using versions and adaptation patterns', *International Conference on Research Challenges in Information Science (RCIS)*, May, Greece, pp.400–411.

Brocke, J., Zelt, S. and Schmiedel, T. (2016) 'On the role of context in business process management', *International Journal of Information Management*, Vol. 36, No. 3, pp.486–495.

Chaâbane, M.A., Andonoff, E., Bouaziz, R. and Bouzguenda, L. (2009) 'Versions to address business process flexibility issue', *International Conference on Advances in Databases and Information Systems*, September, Riga, Latvia, pp.2–14.

Dey, A.K., Abowd, G.D. and Salber, D. (2001) 'A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications', *Journal of Human Computer Interaction,* Vol. 16, No. 2, pp.97–166.

Dumas, M., van der Aalst, W. and ter Hofstede, A. (2005) *Process-Aware Information Systems: Bridging People and Software Through Process Technology*, Wiley & Sons, Hoboken, New Jersey.

Ekanayake, C., La Rosa, M., Ter Hofstede, A.H. and Fauvet, M.C. (2011) 'Fragment-based version management for repositories of business process models', *International Conferences OTM Confederated On the Move to Meaningful Internet Systems Hersonissos*, Crete, Greece, pp.20–37.

Ellouze, F., Chaabane, M.A., Bouaziz, R. and Andonoff, E. (2016) 'Addressing inter-organisational process flexibility using versions: the VP.2M approach', *International Conference on Research Challenges in Information Science*, June, Grenoble, France, pp.1–12.

Ellouze, F., Chaâbane, M.A., Andonoff, E. and Bouaziz, R. (2017) 'Onto-vP.2M: a new approach to model and manage collaborative process versions using contexts and ontologies', *International Journal of e-Collaboration,* Vol. 13, No. 3, pp.39–62.

Hallerbach, A., Bauer, T. and Reichert, M. (2010) 'Capturing variability in business process models: the provop approach', *Journal of Software Maintenance and Evolution: Research and Practice,* Vol. 22, Nos. 6–7, pp.519–546.

Hoang, H., Jung, J. and Tran, C. (2014) 'Ontology-based approaches for cross-enterprise collaboration: a literature review on semantic business process management', *International Journal of Enterprise Information Systems,* Vol. 8, No. 6, pp.648–664.

Jin, T., Wang, J., La Rosa, M., Ter Hofstede, A. and Wen, L. (2013) 'Efficient querying of large process model repositories', *Journal of Computers in Industry,* Vol. 64, No. 1, pp.41–49.

Kumar, A. and Yao, W. (2012) 'Design and management of flexible process variants using templates and rules', *Journal of Computers in Industry,* Vol. 63, No. 2, pp.112–130.

La Rosa, M. and Dumas, M. (2008) *Configurable Process Models: How To Adopt Standard Practices In Your How Way?*. BPTrends Newsletter.

Lassoued, Y., Bouzguenda, L. and Mahmoud, T. (2016) 'Context-aware business process versions management', *International Journal of e-Collaboration,* Vol. 12, No. 3, pp.7–33.

Lu, R., Sadiq, S. and Governatori, G. (2009) 'On managing business processes variants', *International Journal of Data and Knowledge Engineering,* Vol. 68, No. 7, pp.642–664.

Natschläger, C., Geist, V., Illibauer, C. and Hutter, R. (2016) 'Modelling business process variants using graph transformation rules', *International Conference on Model-Driven Engineering and Software Development*, February, Rome, Italy, pp.65–74.

Nie, H., Lu, X. and Duan, H. (2014) 'Supporting BPMN choreography with system integration artefacts for enterprise process collaboration', *in International Journal of Enterprise Information Systems,* Vol. 8, No. 4, pp.512–529.

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, pp.378–387.

Nurcan, S. and Edme, M.H. (2005) 'Intention driven modelling for flexible workflow applications', *International Journal on Software Process: Improvement and Practice,* Vol. 10, No. 4, pp.363–377.

OMG (2014) *Business Process Model and Notation (BPMN) Version 2.0*, http://www.omg.org/spec/BPMN/2.0

Polyvyanyy, A., La Rosa, M. and Ter Hofstede, A.H. (2014) 'Indexing and efficient instance-based retrieval of process models using untanglings', *International Conference on Advanced Information Systems Engineering*, Thessaloniki, Greece, pp.439–456.

Polyvyanyy, A., Ouyang, C., Barros, A. and van der Aalst, W.M. (2017) 'Process querying: enabling business intelligence through query-based process analytics', *Journal of Decision Support Systems,* Vol. 100, pp.41–56.

Reichert, M. and weber, B. (2012) *Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies*, Springer, Heidelberg, New York Dordrecht London.

Reijers, H. (2006) 'Workflow flexibility: the Forlon promise', *International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Manchester, UK, pp.271–272.

Rolland, C. (2010) 'Fitting system functionality to business needs: alignment issues and challenges', *International Conference on Software Methodologies,* Tools and Techniques, Yokohama City, Japan, pp.137–147.

Rosemann, M. and Recker, J. (2006) 'Context-aware process design exploring the extrinsic drivers for process flexibility', *The International Conference on Business Process Modeling, Development and Support*, Luxembourg, pp.149–158.

Rosemann, M., Recker, J. and Flender, C. (2008) 'Contextualisation of business processes', *International Journal of Business Process Integration and Management,* Vol. 3, No. 1, pp.47–60.

Saidani, O. and Nurcan, S. (2009) 'Context-awareness for adequate business process modelling', *International Conference on Research Challenges in Information Science*, Fes, Morocco, pp.177–186.

Saidani, O., Rolland, C. and Nurcan, S. (2015) 'Towards a generic context model for BPM', *Hawaii Conference on System Sciences*, Kauai, Hawaii, 4120–4129.

Santos, E., Pimentel, J., Castro, J., Sánchez, J. and Pastor, O. (2010) 'Configuring the variability of business process models using non-functional requirements', *Conference on Business Process Modelling, Development and Support*, Hammamet, Tunisia, pp.274–286.

Scheer (2015) *Scheer BPaaS*, Retrieved from https://aws-institut.de/wp-content/uploads/2016/02/AWScheer_Whitepaper5_Industrie-4_0.pdf

Sheng, Q.Z. and Benatallah, B. (2005) 'ContextUML: a UML-based modelling language for model-driven development of context-aware web services', *International Conference on Mobile Business*, Sydney, Australia, pp.206–212.

Signavio (2015) https://docs.signavio.com/userguide/editor/de/index.html

Strang, T. and Linnhoff-Popien, C. (2004) 'A context modelling survey', *Workshop on Advanced Context Modelling,* Reasoning and Management, Nottingham, UK.

Ter Hofstede, A.H., Chun, O., La Rosa, M., Song, L., Wang, J. and Polyvyanyy, A. (2013) 'APQL: a process-model query language', *Asia Pacific Conference on Business Process Management (AP-BPM)*, Beijing, China, pp.23–38.

van der Aalst, W. (2013) 'Business process management: a comprehensive survey', *Journal on Software Engineering,* doi: 10.1155/2013/507984, pp.1–37.

Wang, J., Jin, T., Wong, R.K. and Wen, L. (2014) 'Querying business process model repositories', *Journal of World Wide Web,* Vol. 17, No. 3, pp.427–454.

Wang, X.H., Zhang, D.Q., Gu, T. and Pung, H.K. (2004) 'Ontology based context modelling and reasoning using owl', *Workshop on Pervasive Computing and Communications*, Orlando, FL, USA, pp.18–22.

Weske, M. (2007) *'Business Process Management: Concepts, Languages, Architectures*, Springer, Potsdam, Germany.

Zhao, X. and Liu, C. (2013) 'Version management for business process schema evolution', *International Journal of Information Systems*, Vol. 38, No. 8, pp.1046–1069.

Appendix: BPMN4V-QL Grammar

```
BPMN4VQuery : selectQuery ;
selectQuery  :SELECT versionableConcept
              WHERE predicate
             MATCHING matchingType EOS;
VersionableConcept      :
VersionableConceptName LP UNIVERSE|
ConceptName RP;
versionableConceptName : Version_of_Task'
            |'version_Of_Process'
                  |'Version_of_Collaboration'
                  |'Version_of_Event'
                  |'Version_of_Resource'
                  |'Version_of_ResourceRole'
                  |'Version_of_ItemAwareElement'
                  |'Version_of_Message';
matchingType    :'Exact'|'Subset'|'Superset';
predicate          :proposition|
                    setOfPropositions|
                   componentProposition|
                   setOfComponentPropositions
                    logicalTest;
logicalTest        :isTrue||isFalse;
setOfPropositions : (proposition) EOS (proposition)
(EOS proposition)*
proposition        :  setPredicate | negation;
negation           : NOT proposition;
setPredicate       : contextElement situation
(pertinenceDegree)?
situation: Equals|Subsume|In;
Equals     : 'equals' Value
Subsume :'subsume' Value
In         : 'in' LSB Value EOS Value RSB
Value    :STRING|INT|NUMBER;
pertinenceDegree : 'High'|'Medium'|'Low';
componentProposition   : CONTAINS
versionableConceptName ConceptName WHEN
setPredicate
setOfComponentPropositions :
(componentProposition) EOS (componentProposition)
(EOS componentProposition)*
isTrue       : proposition IS TRUE ;
isFalse : proposition IS FALSE;
contextElement  :STRING;
ConceptName :STRING;
UNIVERSAL    : '*' ;
STRING : DQ ('a'..'z'|'_') ('a'..'z'|0..9|'_')* DQ;
INT         : (0..9)*;
NUMBER    : (0..9)*.(0..9)*;
SELECT      : 'SELECT' ;
WHERE       : 'WHERE' ;
MATCHING : 'MATCHING' ;
CONTAINS: 'CONTAINS'  ;
WHEN: 'WHEN' ;
TRUE          :'TRUE';
FALSE         :'FALSE';
IS                  :'IS';
EOS                 :';' ;
LP                  :'(';
RP                  :')';
DQ                  :'"';
LSB                 : '[' ;
RSB                 : ']' ;
```