VerFlexFlow and Querying Language for Business Process Model

Mohamed Amine CHAÂBANE, Lotfi BOUZGUENDA, Rafik BOUAZIZ
Mir@cl laboratory/ University of Sfax/ISIM
Route de Tunis, km 10.
BP 242, 3021 Sakeit Ezzit, Sfax-Tunisia
{MA.Chaabane, Rafik.bouaziz}@fsegs.rnu.tn, lotfi.bouzguenda@isimsf.rnu.tn

ABSTRACT: *Business process flexibility now becomes a crucial research area. In the current study we provide a Versioned Business Process meta-Model (VBP-M for short) for describing flexible business processes. This meta-model is organized around five perspectives (functional, process,informational, organizational and operation). Second, it proposes a language to define, manipulate and query the business process versions conforming to the proposed VBP meta-model. Finally, this paper shows an overview of the VerFlexFlow tool that we propose to validate our contributions.*

Keywords: Business Process, Flexibility, Versions and VerFlexFlow

Received: 29 December 2010, Revised 28 January 2011, Accepted 4 February 2011

1. Introduction

Recently, organizations are interested to development of information and communication technology to automate their Business Processes (BP) and adopt a workflow management system to realize them. In spit of the important advances have been done in BP management, some effectiveness of BPs in information system are not yet achieved and several challenging issues are still to be addressed. One of the most important issues is the business process flexibility [1].

Indeed, nowadays, organizations are competitive, this competition leads them to usually change and adapt their BP to meet new organizational and customer requirement. Currently, researchers in business process management are widely interested in business process flexibility. Literature provides several definitions of business process flexibility. For instance in [2], flexibility is defined as the ability to deal with both foreseen and unforeseen changes in the environment in which business processes operate. In [3], flexibility is defined as the capacity of making a compromise between, first, satisfying rapidly and easily the business requirements in terms of ability when organizational, functional and/or operational changes occur, and second keeping effectiveness. So far, despites the efforts of the Business Process Management community, there is not yet an agreement on BP flexibility.

Our previous research works defend the use of the version concept to address the BP flexibility. In fact, versions are used in several fields of computer science in which is highlighted the need to describe evolution of entities over time. Thus, versions are used in databases [4], [5] in software engineering to handle software configurations [6] and in conceptual models such as the Entity Relationship or the OMT models [7] [8].

Some efforts have also been put on version management in the business process context, and partial solutions to BP version modeling and instance adaptation and migration have been proposed in the literature. These solutions have in common the adoption of an activity-oriented based approach to design BP. Proposed solutions define a set of operations supporting both

business process schema (definition) change, and adaptation and migration of their corresponding instances ([9], [10], [11]). Especially, we have shown in [12] how versions are useful to deal with some types of both a priori and a posteriori flexibility according the two main taxonomies proposed in ([2], [3]). More precisely, we have proposed in [12] a BP metamodel based on versions for describing business process versions.

The problem being addressed in this paper is "how to define, query and manipulate business process versions modeled with this meta-model".

This paper recalls, in the first hand, the versioned metamodel we propose to consider BP versions to address the flexibility issue, then it proposes, in the other hand, a language to facilitate the defining, querying and manipulating processes versions designed according the proposed meta-model.

The remainder of this paper is organized as follows. Section 2 exposes some related works. Section 3 presents briefly the Versioned Business Process (VBP) meta-model that we provide for flexible business process designing. Section 4 presents the language that we propose for defining, manipulating and querying versions of BP. Section 5 gives an overview of our VerFlexFlow Tool. Finally, section 6 concludes the paper and gives our future work.

2. Related Works

The problem of flexibility has mainly been addressed in the business process context using two main approaches: an adaptive-based approach and the version-based approach. The adaptive-based approach consists in defining a set of operations supporting both workflow process schema change, and adaptation and migration of their corresponding instances ([9], [10], [11], [13]). In this approach, only one schema is kept for all modelled workflow processes. This approach has been investigated intensively and the ADEPT Workflow Management System (WfMS) [13] is probably the most successful WfMS regarding workflow process' schema evolution.

In the version-based approach, different instances of a same workflow process can have different schemas. Thus, it is possible to distinguish between temporary and permanent updates for workflow processes since it is possible to keep track of chronological workflow process changes, each one representing a possible schema for the considered workflow process.

In the workflow context, where long-term processes are involved, adaptation and migration of workflow process instances according to a new schema are not always easy and are sometimes impossible [9]. So, it is important to be able to manage different schemas for a workflow process in order to allow several instances of this workflow process to own different schemas [14]. Thus, the version-based approach is a promising solution to deal with business process evolution.

Although versions are used in several areas of computer science, to the best of our knowledge, only few efforts have been put on version management in the business process (workflow) context (in the remainder of the paper, the terms workflow and business process will be used equally).

We distinguish two main contributions about versions of business processes in literature. [14] have proposed to deal with dynamic workflow evolution, i.e. modification of workflow process schemas in the presence of active workflow process instances, introducing versions of workflow process schemas. This work has defined a set of operations for workflow process schema modification and, if possible, a strategy for migration of workflow process instances. [15] have also defended the advantages of a version-based approach to face business process evolution. More precisely, this work proposes to model versions of workflow process schemas using graphs. It also presents a set of operations enabling updates of graphs and defines two strategies to extract versions of workflow process schemas from these graphs.

We believe that these two propositions need to be revisited. Indeed, both [14] and [15] addressed the issue of business process versioning only considering two perspectives: functional and processes. These perspectives describe activities involved in the process and their coordination. But, using only these perspectives is not enough to have a comprehensive description of business processes [16]. Three others perspectives have to be considered: the organizational, the informational and the operation perspectives. The organizational perspective structures the business process actors and authorizes them, through the notion of role, to perform tasks making up the process. The informational perspective defines the structure of the documents and data required and produced by the process. These two perspectives are glued together with the process perspective since, in addition to the tasks and their coordination, the process perspective also defines the required resources (information, actors) to perform the tasks. The operation perspective describes elementary operations performed by actors involved in the process.

Consequently, in our works we propose to revisit the business process flexibility problem using a version-based approach and considering the five perspectives: functional, process, operation, organizational and informational perspectives of business processes.

3. Versions To Model Flexible Business Process

This section briefly introduces the Version Business Process (VBP) meta-model we propose to model versions of business processes. It only focuses on the main concepts of the meta-model first introducing the notion of version, and then presenting the VBP meta-model for BP versioning.

3.1Version Concept

As illustrated in figure 1 below, a real world entity has characteristics that may evolve during its lifecycle: it has different successive states. A version corresponds to one of the significant entity states. So, it is possible to manage several entity states (neither only the last one nor all the states). The entity versions are linked by a derivation link; they form a version derivation hierarchy. When created, an entity is described by only one version. The definition of every new entity version is done by derivation from a previous one. Such versions are called derived versions. Several versions may be derived from the same previous one. They are called alternative versions.

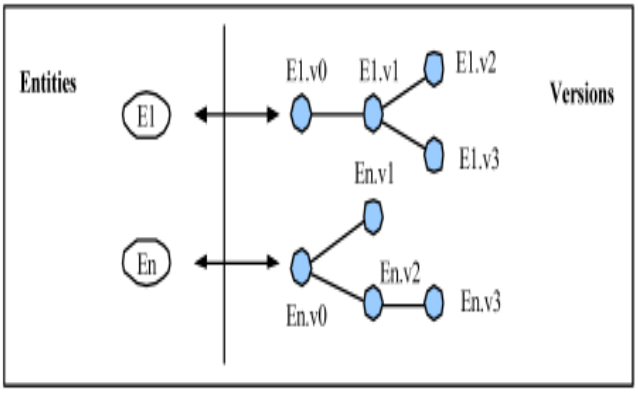

Figure 1. Versions to describe Entity Evolution

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

3.2 The Versioned Business Process meta-model

The Versioned Business Process (VBP) meta-model shown in figure 2 is the result of fusion of two layers: (i) BP metamodel considering the classical BP concepts and (ii) a versioning kit helping to make some concepts (classes of the meta-model) versionnable i.e. classes which we would like handle versions. Because of space limitation we only present, in this paper the VBP meta-model. The interested reader can refer to the [12] [17], [18].

The main concepts of the meta-model are the Process, Activity, Control Pattern, Operation, Informational Resource, and Role concepts. These concepts are organized around the five perspectives described previously. In the functional perspective we describe how a composite activity is decomposed by atomic or composite activities. In the process (or control flow) perspective, execution conditions (preconditions and post-conditions) and the coordination between activities (control pattern) are specified. Generally, the functional perspective and the process perspective are given by the process definition. The operational (or application) perspective defines elementary operations performed by atomic activities. Typically, these operations are used to create, read or modify control and production data, which are often, executed using external applications. The organizational (or resource) perspective describes relationships between roles, groups and actors giving these latter authorizations to perform atomic activities. Finally, the informational (or data) perspective deals with production and use of information. We can note that these perspectives have in common classes; for instance the Atomic activity class both belongs to the process and the functional perspectives.

The underlying idea of our proposition to take into account versions of business processes is to describe, for each versionable class (presented with a grey colour), both entities and their corresponding versions as indicated in figure 1. As a consequence, each versionable class is described using two classes: a first class, called "…", to model entities and a second one, called "Version_ of_…", whose instances are the corresponding versions. For instance, versions of processes are modelled within two classes: the Process class gathers the different modelled business process while the Version_Of_Process class gathers the different versions of the modelled processes. These classes are linked together by two relationships: the "Is_version_of" relationship links a versionable class with its corresponding "Version of…" class and the "Derived_from" relationship describes version derivation hierarchies between versions of a same entity. This latter relationship is reflexive and the semantic of both sides of this relationship are: (i) a version (SV) succeeds another one in the derivation hierarchy and, (ii) a version (PV) precedes another one in the derivation hierarchy.

The VBP meta-model distinguishes six versionable classes. Regarding the process perspective, we propose to keep versions for only three classes: the Atomic Activity, the Operation and the Process classes. It is indeed interesting to keep changes history for atomic activities, operations and processes since these changes correspond to changes in the way that business is carried out. At the atomic (activity or operation) level, versions describe evolutions in activity realization while at the process level; versions describe evolutions in work organization (i.e. coordination of activities).

Regarding the other perspectives, it is necessary to handle versions for the informational resource class from the informational perspective, and versions for the Role Organizational Unit classes from the organizational perspective.

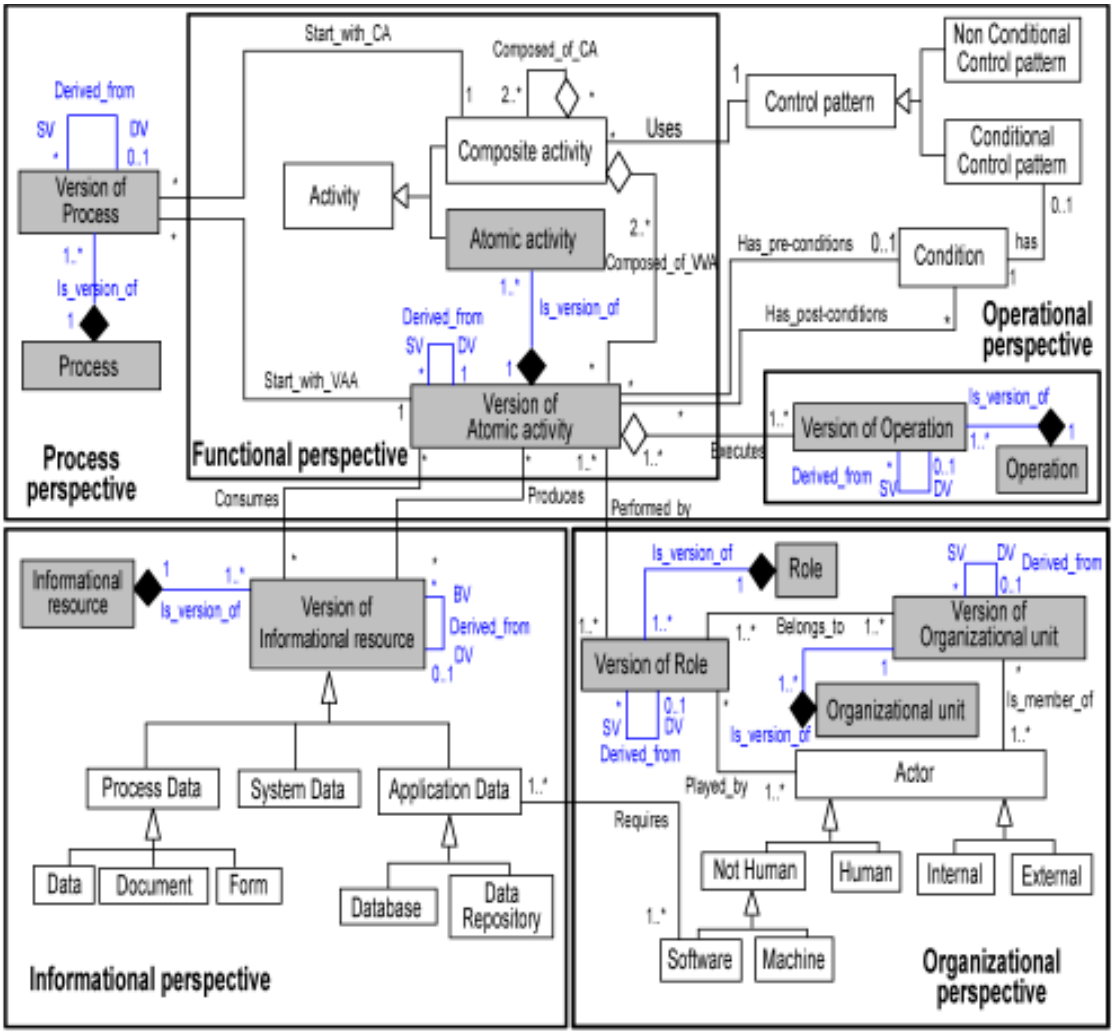

Figure 2. The versioned BP meta-model

4. A Business Process Versions Querying And Manipulating Language

The VBP meta-model presented in section 3, describes the basic workflow entities and their relationships to consider BP flexibility. To facilitate the manipulation of the versions of BP and extract information from this meta-model, it is necessary to define operations addressing versions of BP.

These operations should have clear syntax and unambiguous semantics. They need to be coherent and complete to manipulate versioned BP.

The figure 3 gives an UML state chart which indicates when these operations are available. Some of them are available whatever the states of the versions on which they are applied, while others are only available in some cases. In this state chart, each operation is described using the notation Event/Action whose meaning is "if Event appears then Action is triggered".

When the create-order event is satisfied, the create action is performed to both create the entity and its corresponding first version. The state of the created version is working. In this state, the version can be updated or deleted in a destructive manner; with out keeping track of changes. If deleted, a version reaches its final state of the chart.

When a working version becomes stable, i.e. it does not need additional updates, a freeze-order event can appear to trigger a freeze action. Its state is then frozen (long-lasting), so it can be used in or as a VBP. However, a frozen version can be deleted, but in not destructive manner (to keep track of the process evolution), or can serve as a basis for the creation of a new version using the derive operation. In this case, the derive action create the new version whose state is working and a new life cycle is beginning. A deleted version can be vacuumed when its history becomes not significant.

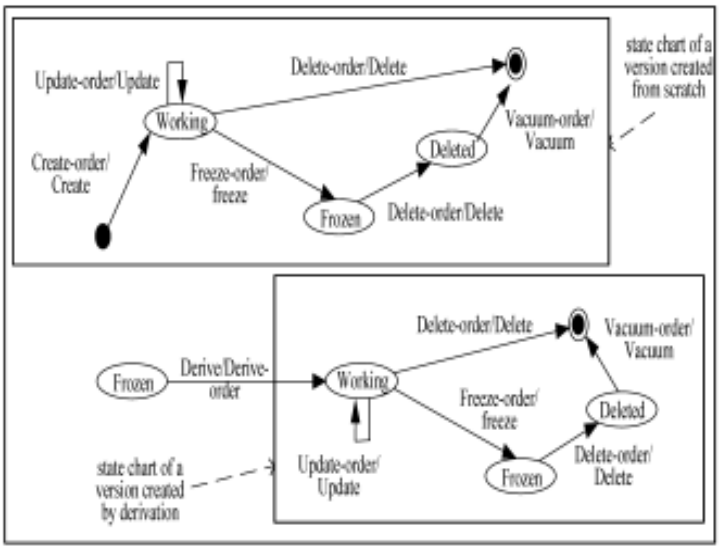

Figure 3. State chart of BP Version

This state chart gives just a general idea of possible actions and their effects, the operations ensuring these actions require further details. In the remainder of this section we explain the syntax of each operation respecting the Backus-Naur Form (BNF).

4.1 Versions Definition Operations

1) *Create operation*.

We distinguish for versionable class a specific syntax of the create operation. Generally, a create operation creates a new object into a versioned class and of its first version. For a Create operation, some information's should be defined by the designer (e.g. name of the created object) and others complementary information's are added automatically such as the identifier of process, the version number, the creation date. Because of space limitation, we only detail the two most important operations e.g create process and create atomic activity.

Syntax:

```
CREATE PROCESS (name_process, description_process,creator_name),
STRAT WITH
(VAA=<id_version_atomic_activity>|CA=<id_composed_activity>);
```

b. *Create atomic activity* allows creating a new atomic activity. An atomic activity has precondition and postconditions, executes operations, invokes role and consumes and produces informational resources. The Create atomic activity operation is performed only if the component elements of the new created activity are created before. When create an atomic activity the designer must give a name for the creating atomic activity, a short description, the name of the creator, its precondition, its post-conditions, its consumed informational resources, its produced informational resources, its operations and roles which invoked in its execution.

Syntax:

```
CREATE ATOMIC_ACTIVITY (name_atomic_activity,
description_atomic_activity, creator_name,
[HAS_PRECONDITION (ID=<id_condition>),
[HAS_POSTCONDITION
(ID=<id_condition>(,<id_condition>)),]
[CONSUMES_INFORMATION(ID=<id_version_informtion>(,
<id_version_information>)),]
[PRODUCES_INFORMATION(ID=<id_version_information>(
,<id_version_information>)),EXECUTE_OPERATION(NAME
=<name_operation> (,<name_operation)),
PERFORMED_BY_ROLE (ID=<id_version_role> (,
<id_version_role>));
```

2) Derive Operation

The derive operation is enabled only for the frozen versions this operation creates a new derived version. This new derived version has working status. Moreover, the derivation of a versionable class can be propagated to others versionables classes according the derivation hierarchy presented in figure 4.

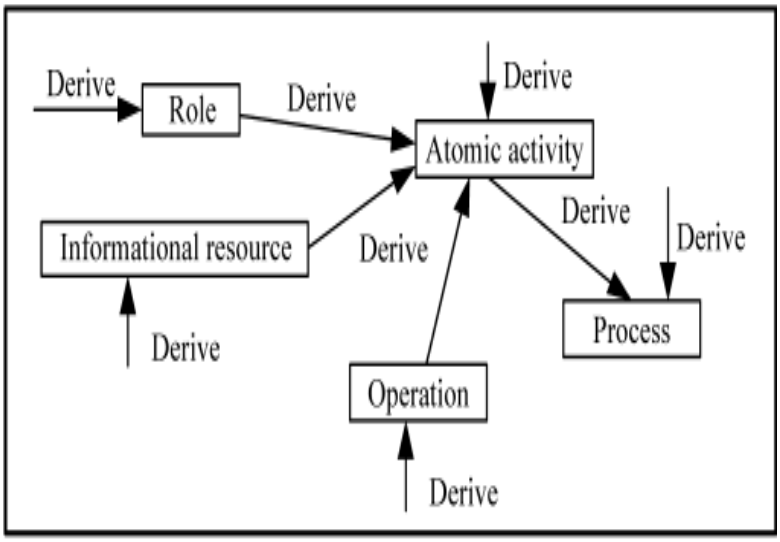

Figure 4. Derivation propagation

The derivation of an operation version, an informational resource version, a Role version or an organizational unit triggers the derivation of its corresponding atomic activity version. In the same way, derivation of *atomic* activity version triggers the derivation of its corresponding process version. Table 1 gives an extract of the semantics of two operations (Derive and Update) according to the classes in which they are defined. Only two versionable classes are considered.

Syntax:

```
DERIVE <versionnable_class>
FROM
(<id_version>|<creator_name>|<creation_date>);
```

4.2 Versions Manipulation Operations

In addition to the previous presented operations, we also propose specific operations to manipulate version such as freeze, update and delete version operation.

1) Freeze operation

Freeze operation allows modifying the state of version from working to freeze if this version have a stable state and don't

Process	**Atomic activity**
1. Change structure	1 Change conditions
1.1 add/delete a composite activity in structure of process	1.1 add/delete pre-conditions (has-pre-conditions relationship
1.2 add/delete atomic activity in structure of process	1.2 add/delete post-conditions (has-post-conditions relationship)
2. change pattern.	2. change operations (executes relationship)
2.1 choose a pattern for its composite activity (uses relationship)	3. Change information
	3.1. add/delete input information (consumes relationship)
	3.2. add/delete output information (produces relationship)
	4. Change role (references relationship)

Table 1.Taxonomy of Derive and Update Operations

endure any transformation. The six classes "Version of…" classes are considered.

Syntax:

FREEZE <versionable_class>
[WHERE
(<id_version>|<creator_version>|<creation_date>)];

2) Update operation

Update operation allows modifying a working version without keeping track the previous version. For a versioning class, the update operation we propose have nearly a same syntax than the create operation but it don't create a new object into "version_of…" class. It only modifies value of an existing version. Table 1 presented before summarizes all scenarios to modify version of process or an atomic activity. Because space limitation we give only the syntax of the update operation for atomic activity version.

Syntax:

```
UPDATE ATOMIC ACTIVITY WHERE
(ID=<id_version_atomic_activity>),
SET [HAS_PRECONDITION (ID=<id_condition>),]
[HAS_POSTCONDITION(ID=<id_condition>{
```

```
,<id_condition>}),
[CONSUMES_INFORMATION (ID=<id_information_vers>
{,<id_information_vers>}),]
[PPRODUCE_INFORMATION (ID=<id_information_vers>
{,<id_information_vers>}),]
EXECUTE_OPERATION
(NAME=<name_operation>{,<name_operation> }),
PERFORMED_BY_ROLE (ID=<id_role_version>) {,
id_role_version});
```

3) Delete operation

Delete operations allows deleting with a non destructive manner working or frozen versions; that mean its state become deleted with out deleting it from the database.

Syntax:

DELETE <verssionnable_class>
WhHERE (<id_version> | <Creator_name> |
<create_date>)

4.3 Versions Querying operation

In addition to definition operation and manipulation operation we propose the interrogation operations, such as Select and Display.

1) Select operation

This operation allows the selection of versions verifying condition. Moreover, the selection of a version may trigger the selection of others versions, which are linked with the selected one (called related version). Figure 5 below illustrates this selection propagation hierarchy.

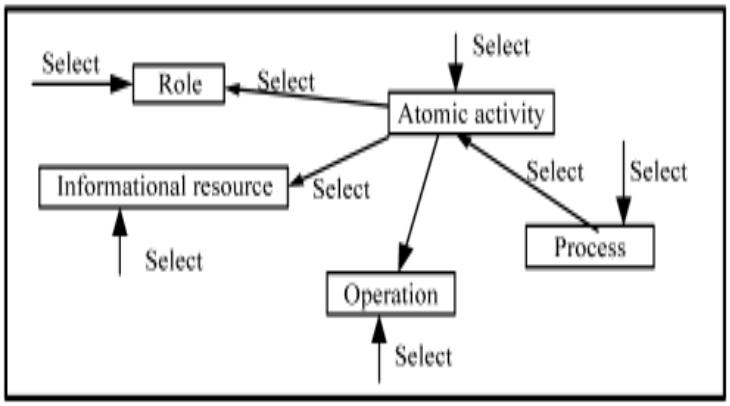

Figure 5. Selection propagation

Indeed, the selection of a process version triggers the selection of all of its activities (atomic or composite) versions. In the same way, the selection of an activity triggers the selection of its invoked role version, its operations and it's consumed and or produced informational resource version.

Syntax:

```
SELECT <versionnable_class>,
[WHERE <id_version>];
```

2) Display operation

Sometimes the results of the selection operation is too complex than the designer needs. The display operation refines this result. Indeed, users can choose only the classes that he need according the hierarchy of the selection propagation presented in figure 5.

Syntax:

```
DISPLAY (<related_class> | all) WHERE
(<name_class> and <id_ver_class>);
```

5. Implementation

This work has been implemented as a part of the VerFlexFlow Design Project, whose objective is to provide a frame work to design and specify Business Process Versions according to the VBP meta-model. VerFlexFlow $_{Design}$ has been implemented with Eclipse platform. It provides two manners for defining, querying and manipulating Business Process Versions:

• VerFlexFlow$_{forms}$ gives some forms to assist the designers to manage BP versions,

• VerFlexFlow$_{language}$ implements the proposed language.

• VerFlexFlow$_{Graphic}$ gives a graphical specification using BPMN (Business Process Modeling Notation) for Business process versions designed according the VBP meta-model [18] [19].

Because of space limitation, we only give here the interface corresponding to the VerFlexFlow$_{language}$. Figure 6 shows the VerFlexFlow Language interface.

More precisely, this interface is organized around two parts: the first one allows the designer to write its operations for BPs versions management like create, derive, update and so on. The second part allows the visualization of the result of the query. For instance, this figure illustrates how we create an atomic activity according to the syntax described previously.

Figure 6. A VerFlexFlow$_{language}$ screenshot

6. Conclusion

This paper has briefly presented a versioned BP metamodel to deal with business process flexibility. In order to manipulate versions of BPs modelled using the VBP metamodel, we have defined a Business Process Versions Querying and Manipulating Language. This language defines taxonomy of operations for BPs versions: it permits to create, derive, update, delete or select BP versions. This paper has presented our VerFlexFlow $_{Design}$ tool which validates our contribution and notably, the VerFlexFlow$_{Language}$ interface.

As future Work, we intend to propose several implementation kits responsible for the derivation of BP versions (instances of the meta-model) onto specific BP languages such as XPDL standard.

7. Acknowledgment

The authors acknowledge all the participants involved in the VerFlexFlow Project development. More precisely, they acknowledge Emna JAMOUSSI, Jihen CHAABANE and Imen BEN SAID.

References

[1] Reijers, H. (2006). Workflow Flexibilty: the Forlon Promise, *In*: Int. Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Manchester, United Kingdom, 271–272.

[2] Schoneneberg, H., Mans, R., Russell, N., Mulyar, N., van der Aalst, W. (2008). Process Flexibility: A Survey of Contemporary Approaches, *In*: Int. Workshop on CIAO/EOMAS at Int. Conference on Advanced Information Systems, Monpellier, France, 16–30.

[3] Nurcan, S. (2008). A Survey on the Flexibility Requirements related to Business Process and Modeling Artifacts. Hawaïï International Conference on System Sciences, Waikoloa, Big Island, Hawaï, USA, 378.

[4] Sciore, E. (1994). Versioning and Configuration Management in Object- Oriented Databases, *Int. Journal on Very Large Databases*, 77–106.

[5] Chen, I., Markowitz, I., Li, P., Fasnan, K (1996). Version Management for Scientific Databases, *In*: Int. Conference On Extended Database Technology, Avignon, France, 289–303.

[6] Kimball, J., Larson,A (1994). Epochs: Configuration Schema, and Version Cursors in the KBSA Framework CCM Model, *In*: Int. Workshop on Software Configuration Management, Trondheim, Norway, 33–42.

[7] Roddick, J., Craske, N., Richards, T (1993). A Taxonomy for Schema Versioning based on the Relational and Entity Relationship Models, *In*: Int. Conference. on the Entity Relationship Approach, Arlington, Texas, USA, 137–148.

[8] Andonoff, E. Le Parc, A., Hubert, G., Zurfluh, G. (1996). Context Aware Business Process Evaluation and Redesign, *In*: Int Conference on the Entity Relationship Approach, Cottbus, Germany, 472–487.

[9] Casati, F., Ceri, S., Pernici, B., Pozzi,. G (1996). Workflow Evolution, *In*: Int. Conference on the Entity Relationship Approach, Cottbus, Germany, 438–455.

[10] Kammer, P. , Bolcer, G., Taylor, R., Bergman, M (2000). Techniques for supporting Dynamic and Adaptive Workflow. *Int. Journal on Computer Supported Cooperative Work*, 269–292.

[11] Rinderle, S., Reichert, M., Dadam, P (2004). Disjoint and Overlapping Process Changes: Challenges, Solutions and Applications, *In*: Int.Conference on Cooperative Information Systems, Agia Napa, Cyprus,101–120.

[12] Chaâbane, M. A., Andonoff, E., Bouaziz, R., Bouzguenda, L. (2009). Versions to Address Business Process Flexibility Issue. In East-European Conference on Advances in Databases and Information Systems (ADBIS 2009), Riga, 07-SEP-09-10-SEP-09, Janis Grundspenkis, Tadeusz Morzy, Gottfried Vossen (Eds.), Springer, p. 2-14.

[13] Reichert,M., Dadam, P (1998). ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control. *Int. Journal on Intelligent Information Systems*, 93–129.

[14] Kradofler,M., Geppert, A (1999). Dynamic Workflow Schema Evolution based on Workflow Type Versioning and Workflow Migration., Int. Conference on Cooperative Information Systems, Edinburgh, Scotland, 104–114.

[15] Zhao, X., Liu, C (2007). Version Management in the Business Change Contex, *In*: Int. Conf. Business Process Management, Brisbane, Australia, 198–213.

[16] van der Aalst, W. M.P., Weske., M., Wirtz, G (2003). Advanced Topics in Workflow Management: Issues, Requirements and solutions, *Journal of Integrated Design and Process Science*, 7 (3) 49-77.

[17] Chaâbane, M. A., Andonoff, E. , Bouzgenda, L., Bouaziz, R. (2009). Towards a Version-Based Approach to Deal with Business Process Evolution (Book chapter), *In*: E-Business and Telecommunications, Communications in Computer and Information Science (CCIS) series, J.Filipe and M.S. Obaidat (Eds): ICETE 2008, Springer verlag. p. 74-88.

[18] Chaâbane, M. A., Andonoff, E. Bouaziz, R., Bouzguenda, L.(2010). Modélisation Multidimentionnelle des versions de Processus. *In*: Ingénierie des Systèmes d'Information, 15 (5) 89-114.

[19] Ben Saïd, I., Chaâbane, M.A., Andonoff, E. (2010). A Model Driven Engineering Approach for Modelling versions of Business Process using BPMN, *In* : Business Information System, p. 254-267.