Addressing Inter-Organisational Process Flexibility using Versions: the VP2M Approach

F. Ellouze, MA. Chaabane, R. Bouaziz
University of Sfax, MIRACL
Route de l'aéroport, BP 1088, 3018 Sfax, Tunisia
{fatma.ellouze, ma.chaabane, raf.bouaziz}@fsegs.rnu.tn

E. Andonoff
IRIT, University of Toulouse 1- Capitole
Rue du Doyen Gabriel Marty, 31042 Toulouse, France
andonoff@univ-tlse1.fr

***Abstract*—Process flexibility has been investigated in depth in the context of intra-organisational processes, but it is still an open issue when processes cross the boundaries of companies. In this paper, we address the modelling of flexible inter-organisational processes using a version-based approach. Indeed, versions are known to be a powerful technique to deal with variability, evolution and adaptation of processes, which are the three main needs of process flexibility. More precisely, this paper presents VP2M (Version of Process Meta-Model), a meta-model supporting the modelling of versions of inter-organisational processes, addressing both static and dynamic aspects of VP2M. It also illustrates process version modelling within the Subsea Pipeline process example.**

Keywords—Inter-Organisational Process; Flexibility; Version; VP2M.

I. INTRODUCTION

Business Process Management (BPM) is an established area to model, analyse, simulate, enact, and supervise business processes [1]. BPM has gained adoption in companies as it is now mature for engineering processes that do not change over time [2]. However, in order to face the dynamic, open and competitive environment in which they operate, companies need to frequently change their processes: processes need to be more flexible. Consequently, the process flexibility issue is investigated in depth in the BPM area [3].

Process flexibility is defined as the ability of processes to respond to changes occurring in their operating environment. According to the taxonomy of [3], these changes are in line with the four following needs of flexibility: (i) *variability*, for representing a process differently, depending on the context of its execution, (ii) *adaptation*, for handling occasional situations or exceptions that have not been necessarily foreseen in process schemas, (iii) *evolution*, for handling changes in processes, which require occasional or permanent modifications in their schemas, and finally (iv) *looseness*, for handling knowledge intensive processes whose schemas are not known a priori and which correspond to non-repeatable, unpredictable, and emergent processes. Such processes require loose specifications.

Process flexibility issue has been investigated in depth in the context of intra-organisational processes and several contributions addressed previously enumerated flexibility needs (*e.g.*, [4–5] addressed flexibility by variability, [6] addressed flexibility by adaptation, [7–8] addressed flexibility by evolution and [9–10] addressed flexibility by looseness).

However, this issue is still open in the context of Inter-organisational Processes (IoPs), where different companies put resources and skills in common, and coordinate their respective processes in order to reach a common goal [11]. In such a context, flexibility may be related to the availability of involved partner processes or to the update of IoP schema, which explicitly defines the partner process coordination [12]. Research efforts about flexibility mainly address process availability in the context of dynamic IoPs. Dynamic IoPs refer to processes where the different partners involved are not necessarily known at design-time, or can evolve at run-time. The provided solutions support finding new partners offering requested services, along with negotiation, contracting and service execution in separate or comprehensive frameworks [10, 13]. On the other hand, flexibility of IoP schemas is rather neglected. It is related to the update of the schema to take into account changes occurring in the environment of partner processes. As this schema models the coordination between partners, the update can be related to partner process schema change, or to the adding or the deletion of a partner: in each case it leads to the reorganisation of the coordination [12, 14].

This paper addresses the modelling of flexible IoPs and it focuses especially on the update of the inter-organisational process schema. It recommends a version-based approach to address this issue and it also takes into account the requirements related to IoP and defined in [15]. On the one hand, the version notion has been introduced to address intra-organisational process flexibility (*e.g.*, [7–8, 16–17]) and more precisely, to address: (i) flexibility by evolution as the different significant changes on processes are modelled within process versions, (ii) flexibility by variability since it is possible to model alternative versions, depending on the context, and (iii) flexibility by adaptation if adaptation can be defined at design-time. On the other hand, we also take into account requirements of [15], recommended for IoP modelling. These modelling requirements derive from a deep analysis of the state-of-the-art and they can be used to evaluate inter-organisational process contributions.

The paper contribution is threefold. Firstly, we propose VP2M, a meta-model for inter-organisational process version modelling that takes into account the requirements of [15]. Secondly, we illustrate the modelling of IoP versions as instances of VP2M within the Subsea Pipeline process. Thirdly, we address the dynamic aspects of IoP versioning introducing operations for version management and we recommend operations for handling views, which have been introduced to take into account requirements of [15].

Accordingly, the remainder of the paper is organised as follows. Section 2 reviews related works, considering process flexibility both in an intra-organisational and in an inter-organisational context. It also defends our approach to deal with this issue. Section 3 presents the VP2M meta-model for modelling flexible IoPs. Section 4 introduces the Subsea Pipeline example and illustrates the modelling of several process versions as instances of VP2M, mainly highlighting the modelling of flexibility by variability and by evolution. Section 5 addresses dynamic aspects of inter-organisational process versioning as well as providing a set of algorithms to deal with views deduction from VP2M instances. Finally, Section 6 provides the conclusions and gives some directions for future research.

II. Related Works

Process flexibility has recently received much attention both in intra-organisational and inter-organisational settings. Different contributions have been made to deal with this issue, and they follow several approaches: activity-driven approach (*e.g.*, [4–9, 16–18]), constraint-driven approach (*e.g.*, [19]), data-driven approach (*e.g.*, [20]), case-driven approach (case handling) (*e.g.*, [21]), intention-driven approach (*e.g.*, [22]), and more recently, social-driven approach (*e.g.*, [23–24]). In this paper, we rather focus on activity-driven process flexibility as activity-oriented models are used in the majority of (service-oriented) process management systems. This section reviews activity-driven process flexibility related works in each of these settings. It also positions our contribution with respect to the state-of-the-art.

A. Intra-Organisational Process Flexibility

We first mention works addressing flexibility by variability using the notion of process variants. A variant is an adjustment at run-time of a base process schema according to the context. We differentiate between (i) behavioural approaches, which define the base process schema as a superset of variants and derive a specific process variant by hiding and blocking process components of the base process schema, and (ii) structural approaches that derive a process variant by applying a set of changes to a base process schema. Specific notations and systems such as C-EPC [4], Provop [5] and vBPMN [25] support process variants.

We also mention works recommending a version-based approach to deal with process flexibility. Indeed, as defended in, *e.g.*, [16], the version notion facilitates the migration of instances from an initial schema to a final one, allowing, if the migration is not possible, two different instances of a same process to run according to two different schemas. In addition, this notion addresses *flexibility by evolution* as the different significant changes on processes are modelled within process versions*, flexibility by variability* since it is possible to model alternative versions, depending on the context, and *flexibility by adaptation* if adaptation can be defined at design-time. Several contributions recommend a version-based approach to address intra-organisational process flexibility considering versioning of different process perspectives: the process perspective [7–8, 12, 16–18, 26], but also the informational and organisational perspectives [12, 16, 26].

We finally mention works addressing flexibility by looseness, introducing late binding and late modelling notions to postpone activity modelling or activity concretisation (*i.e.*, to which concrete operation an activity is linked to) from modelling to execution, depending on the context. The main contribution achieving this flexibility need is presented in [9].

B. Inter-Organisational Process Flexibility

First, IoP is defined as a set of interconnected partner processes running in different organisations [11]. This interconnection may be specified according to the following mechanisms: the capacity sharing, the chained execution, the subcontracting, the case transfer, the extended case transfer and the loosely coupled. We agree with [11, 13] when they argue that loosely coupled IoP, where each partner takes care of a specific part of the process, is the most realistic assumption in the dynamic, open and competitive environment of companies. Moreover, the loosely coupled process requires more flexibility as the involved partners are distributed, heterogeneous and autonomous, and the execution control of the process is distributed. Thus this paper focuses on loosely coupled IoP flexibility.

In addition, [15] has derived a set of modelling requirements for the loosely coupled IoPs from an extensive analysis of the state-of-the-art. These requirements, which can be used for evaluating relevant contributions in loosely coupled inter-organisational process, are the following: support of process abstraction concept, support of efficient process assembly, support of a modelling framework, support of process context modelling, support for process modelling at both the design level and the execution level, and finally, support of the global process information schema. Note that the two first requirements are the most important ones: the first one indicates that we must be able to represent private (*i.e.*, internal) parts of involved processes along with their public (*i.e.*, external) parts, while the second one refers the assembling of these private and public parts. Note also that the last requirement aims to give a comprehensive data and message flow (information) schema.

Regarding loosely coupled IoP flexibility, it has been somewhat overlooked. However, some contributions addressed flexibility by looseness in the context of dynamic loosely coupled IoP recommending solutions to face unavailability of partner process [10, 13]. Regarding the other flexibility needs (evolution, variability and adaptation), which are related to the flexibility of loosely coupled IoP schemas, we mention [27–28], [29–30] and [12].

First, [27–28] addressed change propagation from a partner process towards the processes of the other partners involved in a collaboration or in a choreography. More precisely, they provide a set of algorithms to deal with changes of process schema by adding, deleting, replacing or updating process fragments, but they do not consider changes that can affect messages (*i.e.*, information) exchanged between process partners. Secondly, [29–30] proposed a service-based approach to model the loosely coupled inter-organisational processes by combining processes and SOA. More precisely, they provide high-level patterns for service (adding, removing, substituting services), control flow and interaction adaptation.

Note that these contributions address inter-organisational process evolution but they do not address IoP variability and adaptation. Finally, [12] extends BPMN to model versions of collaborations. Indeed, the notion of collaboration supports the modelling of inter-organisational processes in BPMN. Thus versioning collaboration helps to address inter-organisational process flexibility. In addition, [12] recommends a set of interaction patterns to make IoP schema change easier to perform.

C. Paper Positioning

This paper addresses IoP flexibility combining the notion of version, which is helpful to address process flexibility, and the requirements of [15], which are fundamental features of loosely coupled inter-organisational processes. More precisely, we define the VP2M meta-model to model versions of loosely-coupled IoPs.

Benefits of our approach are the following. Thanks to the notion of version, we address flexibility by evolution, flexibility by variability and flexibility by adaptation if adaptation can be defined at design-time. In addition, the VP2M meta-model is comprehensive and simple as it defines the core (basic) concepts for IoP version modelling, taking into account the five perspectives (process, functional, operational, organisational and informational) that processes have to consider to provide a comprehensive picture on the way people cooperate together within and across companies [11]. It also includes the notion of view to take into account the two first fundamental requirements of [15], distinguishing the IoP view (collective view) from the partner's views (individual views) that can be local, global or mixed. On the other hand, we provide algorithms supporting the assembly of the different views along with the basic operations supporting IoP flexibility through IoP versioning.

With respect to related works previously presented ([27–28], [29–30] and [12]), our approach differs from [12] and [27–28] as the basis of these works is BPMN. For instance, [12] extends the BPMN meta-model for collaborations with the notion of version to model version of collaborations. Our contribution also uses the notion of version to address process flexibility. However, instead of BPMN, we recommend a specific meta-model for IoPs taking into account the requirements of [15], particularly the two first and fundamental requirements on process abstraction and process assembly, along with providing a set of concepts for loosely coupled IoP modelling considering the five main perspectives of processes. Indeed, as illustrated in Table 1 below, BPMN and therefore contributions based on BPMN, insufficiently support process abstraction and assembly. They only model separately public and private processes, and consequently they only provide a global view for the IoP (collaboration) and local views for partners processes (private process). In addition, data is poorly represented and there is no collaborative modelling environment for such processes. Finally, [29–30] only addresses flexibility by evolution with no support of process abstraction and assembly.

Table I evaluates the previous contributions with respect to the requirements of [15] as follows: + (-) respectively means that the requirement is supported (not supported) while +/- means that the requirement is partially supported.

TABLE I. RELATED WORKS EVALUATION W.R.T [15]'S REQUIREMENTS

Requirements	[12]	[27–28]	[29–30]
process abstraction	+/-	+/-	-
process assembly	+/-	+/-	-
collaborative modelling framework	+/-	-	-
context for IoPs	+	-	-
modelling at the design level	+	+	+
modelling at the execution level	-	+	-
global business information schema	+/-	+/-	+/-

To sum up, we recommend the VP2M meta-model to model versions of loosely coupled IoPs. We have chosen to define a new meta-model for IoP version modelling instead of extending BPMN for instance as the BPMN meta-model is very complex. More precisely, it does not focus on the core concepts for inter and intra-organisational process modelling. In addition, it does not provide a comprehensive approach for intra and inter organisational process modelling as intra and inter organisational processes are considered separately, respectively within private processes and collaboration/ choreography processes. Finally, it mixes both concepts for process modelling and graphical representation of these concepts, thus making its understanding difficult. On the contrary, VP2M defines the core concepts for IoP version modelling while taking into account the five process modelling perspectives. Moreover, VP2M is independent from any languages, so we can generate executable and/or graphical specifications from instances of VP2M. Finally, VP2M addresses flexibility by evolution, flexibility by variability and flexibility by adaptation if adaptation can be defined at design-time.

III. MODELLING VERSIONS OF INTER-ORGANISATIONAL PROCESSES: THE VP2M META-MODEL

VP2M supports the modelling of versions of IoPs. First, this section introduces the notion of version and then it presentsVP2M as a UML class diagram for versioning both intra and inter-organisational processes.

A. Version Concept and Versioning Pattern

A version corresponds to one of the significant states an entity (in the context of the paper, a process, an activity…) may have during its life cycle. 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, and are organized in derivation hierarchies. We distinguish two types of derivation: (i) derive for variability, for representing choices according to the context, and (ii) derive for evolution, for representing the evolution of an entity, independently from the context, and thus from any choice.

As shown in Fig. 1, we distinguish derivation (*i.e.*, versioning from) for variability, represented with a dotted line, from derivation for evolution, represented as a solid line. In this figure, we consider four versions for the entity E1: the

first one is E1.1. The other versions are the following: E1.2 is an evolution of E1.1, E1.2.1 is a variant of E1.2 and finally E1.3 is an evolution of E1.2.1.

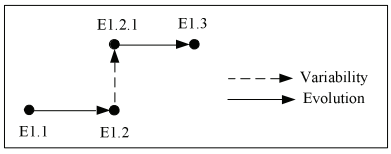

Fig. 1. Derivation by Evolution and Variability

We introduce a versioning pattern to support version modelling. The underlying idea of the versioning pattern is to model, for each versionable class of the VP2M meta-model, both entities and their corresponding versions. According to [16], a versionable class is a class for which we handle versions. As shown in Fig. 2, the pattern is composed of two classes and two relationships. Each versionable class is described as a class, called *Versionable*. We associate to each versionable class, a new class, called *Version_of_Versionable*, whose instances are versions of Versionable, and two new relationships: (i) the *is_version_of* composition, which links each instance of the *Versionable* class with its corresponding instances of the *Version of Versionable* class; and (ii) the *derived_from* , which supports versions derivation hierarchies. This latter relationship is reflexive and the semantics of both relationship sides of *derived_from* is: (i) a version (SV) succeeds another one in the derivation hierarchy and, (ii) a version (PV) precedes another one in the derivation hierarchy. Note that the *derived_from* relationship models both derivation by evolution and derivation by variability.

Regarding versions, we also introduce attributes such as *version number, creator name, creation date* and *state* in the *Version_of_Versionable* class.

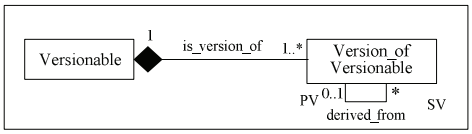

Fig. 2. Versioning Pattern

B. *The VP2M Meta-Model*

This section presents VP2M highlighting the core concepts for both intra and inter-organisational processes modelling. It also introduces the notion of view we add to the meta-model in order to fulfil the main requirements of [15].

VP2M differentiates between two types of process: (i) intra-organisational process, which is modelled as an individual process, namely a process belonging to a single organisation, and (ii) inter-organisational process, which is modelled as a collective process, namely a set of individual processes belonging to different organisations and interacting with one another. We first introduce concepts for intra and inter-organisational process modelling. Then we illustrate their versioning. A UML class diagram of VP2M is given in Fig. 3, focussing on classes and relationships. Fig. 3 adopts the following policies: classes corresponding to versions are visualised in grey, concepts related to individual processes have a blue background, concepts related to collective processes have a yellow background and OCL constraints have a green background.

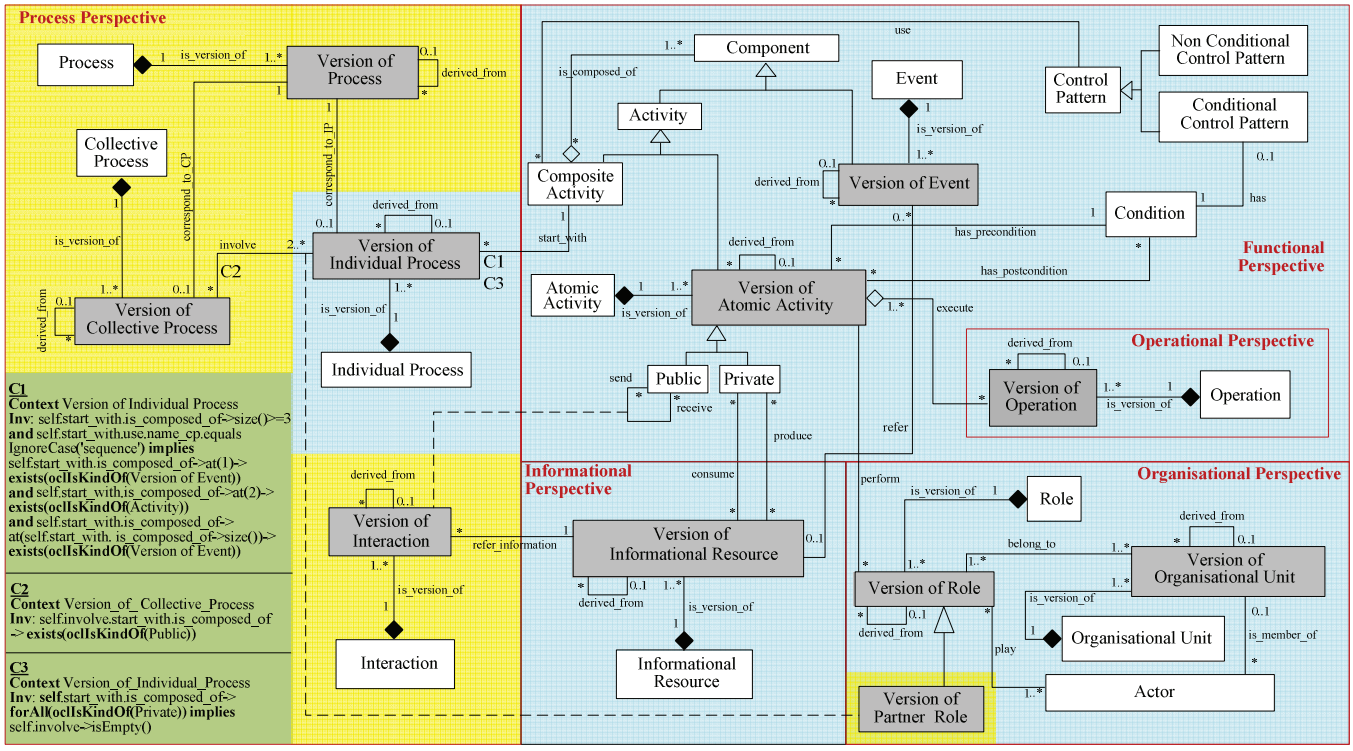

Fig. 3. VP2M: Modelling Versions of Inter-Organisational Process

1) Modelling individual and collective processes. The main concepts for individual processes are Individual Process, Activity, Control Pattern, Event, Informational Resource and Role. We differentiate between composite and atomic activities. A composite activity is a set of components (*i.e.*, activities or events) that are coordinated by control patterns, while an atomic activity refers to a concrete activity, gathering operations (*i.e.*, actions achieved within activities), performed by actors involved in the individual process. Note that an individual process obviously starts with a composite activity having a start event, at least one activity and an end event (*cf.* the OCL constraint C1 in Fig. 3). In addition, an atomic activity has a start condition (precondition), final conditions (post-conditions) and consumes and/or produces informational resources (*i.e.*, data or documents). We also have attributes for components; for instance, the type attribute for an event indicates if it is a start or an end event. Control patterns define the way activities and events are synchronised. They may be conditional (*e.g.*, if, for, while, repeat …) or not (*e.g.*, sequence, fork, join …). Finally, actors involved in the individual process are gathered into organisational units or roles. Note that these concepts support the modelling of the five perspectives of processes, which are known to be essential to have a comprehensive picture of how people work in companies [1–2].

On the other side, we introduce the following concepts to address IoP modelling: *Collective Process*, *Partner Role*, *Public*, *Private*, *Interaction* and *Process*. A collective process defines the set of participating individual processes. Each of these participating individual processes plays a role, denoted as *Partner Role*, in the collective process. We differentiate between public and private atomic activities. Public atomic activities correspond to external atomic activities, namely activities supporting the interaction between participating individual processes, while private atomic activities correspond to internal activities performed by only individual processes. Thus an individual process corresponding to an intra-organisational process is composed of only private atomic activities (*cf.* the OCL constraint C2 in Fig. 3) while an individual process involved in a collective process (*i.e.*, an IoP) is composed of both public and private atomic activities (*cf.* the OCL constraint C3 in Fig. 3). The notion of *Interaction* models the exchange of messages (information) between involved individual processes: we define the source of the interaction (send), the target of the interaction (receive) and the message (information) exchanged between the source and the target (which are public atomic activities). Finally, we introduce the notion of *Process* to model flexible processes changing from individual to collective (or vice-versa). Thus a process may correspond to an individual or to a collective process.

2) Versioning individual and collective processes. We use the versioning pattern previously presented to make some classes of this meta-model versionable. A versionable class is a class for which we manage versions. Thus we differentiate between versionable classes and ordinary classes (*i.e.*, classes for which we do not handle versions). We manage versions for the following classes: *Process*, *Individual Process*, *Collective Process*, *Activity*, *Event*, *Operation*, *Role*, *Partner Role*, *Organisational Unit* and *Interaction*. Indeed, each of these classes represents key concepts for individual and collective processes and plays a strong role in the definition of IoPs. The idea is to keep track of changes occurring to components that play a part in the description of how the IoP is carried out. Note that, unlike the majority of related works (*e.g.*, [7–8, 17–18]), the version notion holds for concepts of both individual and collective processes and it also holds for each perspective of individual processes. As illustrated in Fig. 3, for each of the versionable classes we use the previous versioning pattern and we model both entities and versions. Thus we have two classes for each versionable concept (*e.g.*, Individual Process and Version of Individual Process model individual processes and their corresponding versions).

3) Versioning elements. Generally speaking, a new version of an element (*e.g.*, individual process, activity, event, and collective process) is defined according to changes occurring to it: these changes may correspond to the addition of information (attribute or relationship) or to the modification or the deletion of existing ones. For instance, regarding individual processes, we create new versions when there are changes to the involved activities and/or events or in the way they are synchronised together using control patterns. In the same way, changes to activities and events may result in the creation of new activity and event versions. In addition, we create new versions of public activities involved in message exchange, when there are changes to the exchanged messages. Regarding collective processes, new versions may result from changes to participants involved. Thus when we add or delete a participant, it is necessary to adapt the current collective process to this change: we have to incorporate the added participant or to possibly replace the deleted one. New versions of collective processes may also result from changes to involved individual processes or exchanged messages. Exchanged messages have an important impact in interaction. Thus any change in a sent or a received message affects the involved public activities, and consequently the involved individual process. So, when we add (or delete) a message, we have to add (or to delete) a received and a send public activity, which leads to changing the individual process schema. In this case, the other individual processes involved in the collective process have in turn to be adapted to this change to ensure continued collaboration. Section V will detail operations achieving changes leading to new version definition.

C. Views in VP2M

We previously have defended the importance of the requirements of [15] to address flexible IoPs modelling. We recommend introducing the notion of view to fulfil the process abstraction and process assembly requirements. Indeed, view is known to be a powerful concept to give both internal/private and external/public abstractions of information and it is worth to be used for process abstraction [13, 15, 31–32].

Fig. 4 below gives the different views that can be deduced from instances of VP2M. As for processes, we distinguish collective views from individual views. A collective view holds for collective processes (IoPs). It corresponds to a BPMN collaboration or choreography diagram: only public activities of all the involved individual processes along with their corresponding interaction (exchanged messages) are

visualised. An individual view focuses on a specific partner view, differentiating between local view, global view and mixed view. The first one focuses on the activities of the considered partner while the two others also include public activities of other interacting partners. A local view of a partner extracts only his private and (eventually) public activities. Note that this view holds for both intra and inter-organisational processes, while the two others only hold for inter-organisational processes. On the other side, the global view of a collective process for a partner gathers its public activities along with the public activities of its partners and their corresponding interaction. Finally, the mixed view merges local and global views of a partner, extracting its private and public activities along with public activities of its interacting partners and their corresponding interaction.

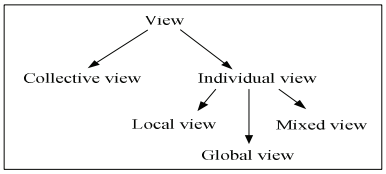

Fig. 4. Views Taxonomy

IV. Modelling Flexible Inter-organisational Processes Using VP2M: The Subsea Pipeline Process

This section illustrates the instantiation of VP2M to model versions of a flexible inter-organisational process, the Subsea Pipeline process from TPS, a petroleum company.

A. Versions of the Subsea Pipeline Process

The Subsea Pipeline process is from TPS, which needs to replace any one of its old damaged subsea pipeline. Three versions of this process are considered in the example. In the first version, the process takes place only within the TPS company. TPS first specifies the necessary team and equipment and then proceeds to the assembling and the welding of pipes on shore by welders and controllers. The next activity is the laying of pipes offshore by the divers. Finally, when the installation is over, a test campaign has to be performed. Note that the assembling and welding, laying and subsea control have to be repeated until reaching the pipeline length.

In the second version of a process, TPS subcontracts the installation of the pipeline to SAROST, a company committed to subsea pipeline installing and maintenance. Thus as shown in Fig. 5, two partners are involved in this process version. TPS first prepares a Tender Specifications (TS) describing the requested pipeline replacement, and submits it to SAROST. Then SAROST carries out a feasibility study and answers either in a positive way sending back to TPS a quote for the pipeline replacement, or in a negative way explaining why it refuses to do the requested job. When the quote is received and accepted by TPS, then it prepares an order for replacement and sends it back to SAROST, which proceeds to the subsea pipeline replacement. After the test campaign, SAROST prepares an acceptance certificate and sends it to TPS, which then ends the process. In the third version, TPS also subcontracts the activity of preparing tender specifications to a consulting office (COff). Thus three partners are involved in this process version: TPS, which is still the initiator of the process, Coff to which TPS sends a request for a tender specification, and SAROST, which replaces the damaged subsea pipeline.

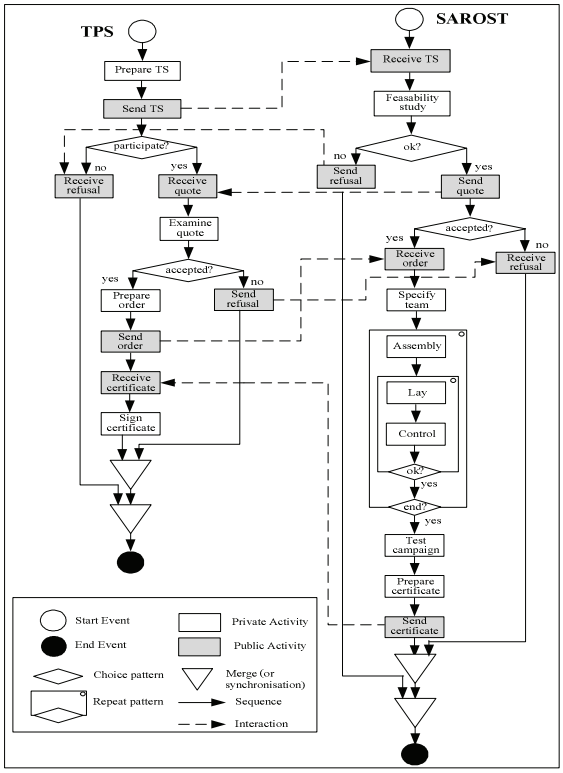

Fig. 5. Versions of the Subsea Pipeline Process

B. VP2M Instantiation

This example considers three versions of the Subsea Pipeline (SP) process. As illustrated in Fig. 6, which is a UML object diagram, these three versions of processes correspond to versions of individual and collective processes. In Fig. 6, relationships from the versioning pattern (*i.e.*, *derived_from* and *is_version_of*) are shown by a black line while the *correspond_to* and *involve* relationships are respectively shown by a blue line and a red line. More precisely, the first version of SP (SP.1) is an intra-organisational process and it refers to the version of the individual process of TPS (TPS-SP.1) containing only private activities (*e.g.*, specify team, assembly, lay).

The two other versions of SP (SP.2 and SP.2.1) are inter-organisational processes as they respectively involve, in addition to TPS, SAROST (in SP.2) and both COff and SAROST (in SP.2.1). They are modelled as two collective processes (CP-SP.1 and CP-SP.1.1), each referring to the individual process of involved partners. Thus we distinguish between three versions of individual processes for TPS (TPS-SP.1, TPS-SP.2, and TPS-SP.2.1), one version of individual process for SAROST (SAROST-SP.1) and one version of individual process for COff (COff-SP.1). The second and third versions of the individual process for TPS and the first version of the individual process of both SAROST (SAROST-SP.1) and COff (COff-SP.1) contain both public and private activities describing respectively interaction between partners

and partner internal actions. In addition, SP.2 is an evolution of SP.1 while SP.2.1 is a variant of SP.2. In the same way, TPS-SP.2 is an evolution of TPS-SP.1 while TPS-SP.2.2 is a variant of TPS-SP.2.1. Finally, the collective process CP-SP.1.1 is a variant of CP-SP.1.

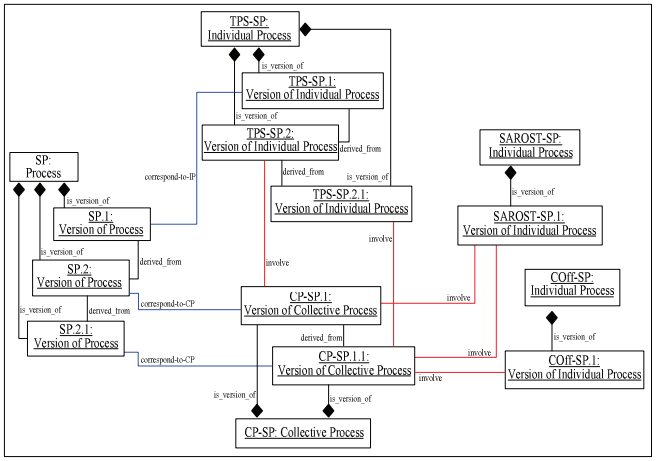

Fig. 6. Partial Instantiation of VP2M for the Subsea Pipeline Process: Process, Individual Process and Collective Process Versioning

Due to lack of space, Fig. 7 details only some of these individual process versions, namely TPS-SP.2 and SAROST-SP.1. SAROST-SP.1 is involved in the collective process version CP-SP.1, which corresponds to the second version of the SP process, namely SP.2. Both TPS-SP.2 and SAROST-SP.1 contain private activities implementing internal actions, and public activities corresponding to the interaction between them. More precisely, Fig. 7 models the following public and private activity versions of TPS-SP.2: PrepareTS (TPS-SP-PrepareTS.1 is the first version of the private activity PrepareTS), SendTS (TPS-SP-SendTS.1 is the first version of the public activity SendTS), ReceiveRefusal (TPS-SP-ReceiveRefusal.1 is the first version of the public activity ReceiveRefusal), ReceiveQuote (TPS-SP-ReceiveQuote.1 is the first version of the public activity ReceiveQuote) and ExamineQuote (TPS-SP-ExamineQuote.1 is the first version of the private activity ExamineQuote). Fig. 7 also models the following public and private activity versions of SAROST-SP.1: ReceiveTS (SAROST-ReceiveTS.1 is the first version of the public activity ReceiveTS), FeasibilityStudy (SAROST-FeasibilityStudy.1 is the first version of the private activity Feasibility Study), SendRefusal (SAROST-SendRefusal.1 is the first version of the public activity SendRefusal), and SendQuote (SAROST-SendQuote.1 is the first version of the public activity SendQuote). Finally, Fig. 7 models the interaction between the public activities of both TPS-SP.2 and SAROST-SP.1: they are visualised in red, blue and brown.

V. VP2M: Dynamic Aspects

This section addresses the dynamic aspects of VP2M introducing operations for version management and algorithms for views deduction. First, it gives a UML state chart diagram indicating when these operations are available. Then, it presents version management operations along with provided algorithms for views deduction.

A. State Chart for Versions

In order to handle versions of IoPs modelled as instances of the VP2M meta-model, we propose a taxonomy of operations which allow the creation, derivation, update, validation, and deletion of versions or which enable or disable versions. The UML state chart given in Fig. 8 indicates when these operations are available with respect to the version state.

According to Fig. 8, a version can be a *Working* version, a *Frozen* version or a *Disabled* version. A working version is a draft version: it can be updated but cannot serve as a support for execution (*i.e.*, it cannot be instantiated). After a series of updates and when it becomes stable, a working version can be validated and therefore it moves to the state *Frozen*. A frozen version describes a significant and stable state of a version, which cannot undergo changes. A frozen version is enabled and can serve as a support for execution (*i.e.*, it can be instan-

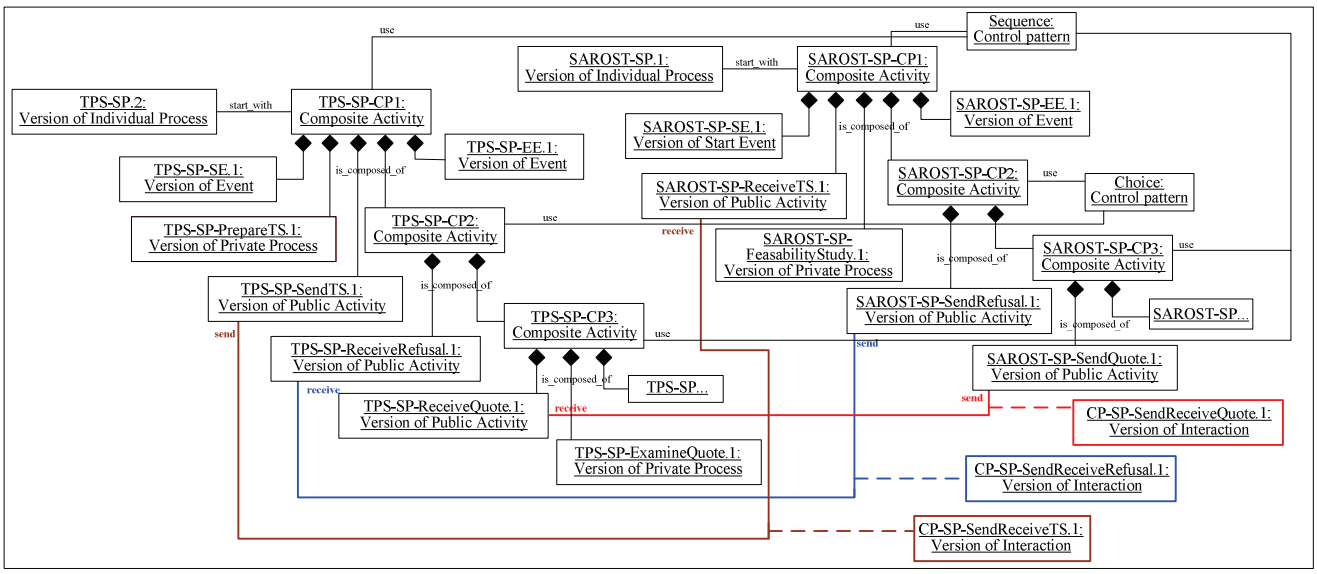

Fig. 7. Partial Instantiation of VP2M for the Subsea Pipeline process: detailing Individual Process Versioning

-tiated). Note that the validation of a version may trigger the validation of other versions (*cf.* section V.B.2). A working version can be deleted while a frozen version can be disabled. It then moves to the state *Disabled*. In this state, the version cannot be instantiated and does not serve anymore as a support for execution. Finally, a stable version can serve as a basis for the creation of a new version using the Derive operation. The new created version is a working version (*cf.* Fig. 8(b)). Before being updated, it has the same value as the derived one. Note that the derivation of a version may trigger the derivation of other versions, which are linked to the derived one.

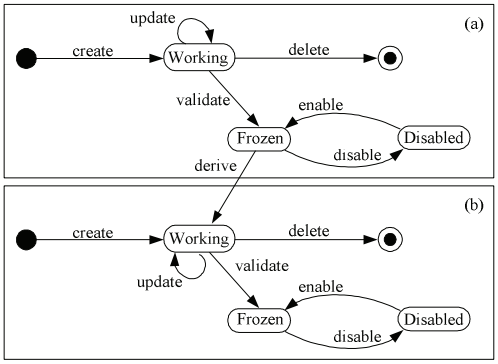

Fig. 8. State Chart for Versions

B. *Operations*

This section details the three main recommended operations, namely the update, the validate and the derive operations.

1) Update Operation. This operation can be specified using a set of primitives that change according to the classes in which they are defined. Table II below summarizes these primitives.

TABLE II. PRIMITIVES OF THE UPDATE OPERATION

Classes	Primitives
Process	+ Individual Process + Collective Process
Individual Process	+/- Activity +/- Event +/- Control Pattern
Collective Process	+/- Individual Process +/- Partner Role
Public Activity	+/- Interaction
Private Activity	+/- Informational Resource +/- Operation +/- Condition +/- Role
Interaction	+/- Send Activity +/- Receive Activity +/- Informational resource
Role	+/- Actor +/- Organisational Unit

For instance, the update of a collective process includes primitives supporting the addition (+) or deletion (-) of involved partners (i.e., their individual processes) and playing a specific role in the IoP. In the same way, the update of an individual process includes primitives supporting the addition or deletion of activities, events and the way they are synchronised.

2) Validate Operation. This operation is performed to make a working version frozen, when the considered version does not need to be updated anymore. Validation of a version may trigger the validation of other versions, which are linked to it via a composition relationship. Fig. 9 illustrates the validation propagation. More precisely, the blue arrows correspond to initial validations while the black arrows correspond to propagated validations. According to Fig. 9, the validation of a version of a collective process triggers the validation of corresponding versions of individual processes. Likewise, the validation of an individual process version triggers the validation of its versioned components, *i.e.* versions of atomic activities and events. In the same way, the validation of atomic activity versions triggers the validation of their linked elements. If this is a public activity, linked interaction, (partner) role and operation versions are in turn validated, while if it is a private activity, linked role, information resource and operation are in turn validated.

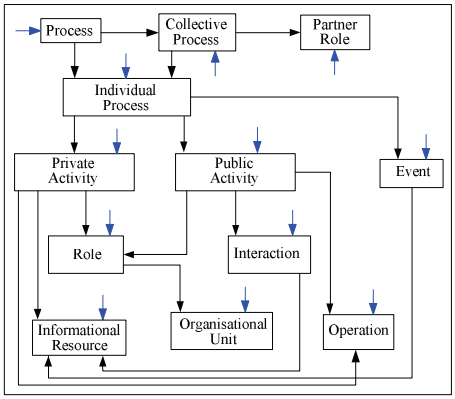

Fig. 9. Validation Propagation

3) Derive Operation. The Derive operation create a new working version as a copy of an existing frozen (*i.e.*, stable) one. Thus before being updated, this new version has the same value as the derived one. Moreover, derivation of a version may trigger the derivation of other versions, which are linked to the derived one. Fig. 10 illustrates this derivation propagation. Again, the blue arrows correspond to initial derivations, while the black ones correspond to propagated derivations.

This propagation is due to the relationships existing between the following classes: Process, Collective Process, Individual Process, Public Activity, Private Activity, Event, Operation, Event, Interaction, Role, Informational Resource and Organisational Unit. Thus the derivation of an individual process or a collective process triggers the derivation of its corresponding process. In the same way, the derivation of an activity (whether private or public) triggers the derivation of its corresponding individual process. In addition, the derivation of an operation, a role and an informational resource triggers the derivation of its corresponding private activity while the derivation of an interaction or a (partner) role triggers the derivation of its corresponding public activity.

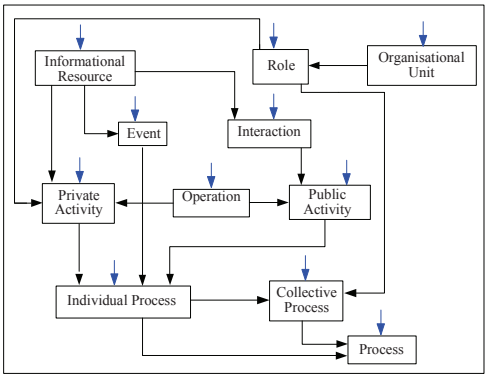

Fig. 10. Derivation Propagation

C. Views Deduction

The notion of view is not explicitly stated in the VP2M meta-model as it can be deduced from private and public activities of individual processes. We introduced a specific pivot structure, the View Data Structure (VDS), for view deduction: instances of VP2M are mapped according to VDS while recommended algorithms deduce views from it. Thus this section firstly provides the recommended mapping algorithms and then those for views deduction. Secondly, it illustrates these mapping and deduction steps within the Subsea Pipeline process.

1) Algorithms for Mapping VP2M onto VDS. As illustrated in Fig. 11, the VDS models VP2M individual processes as trees (IPTree) and VP2M collective processes as graphs (CPGraph), *i.e.,* as set of trees and set of arcs.

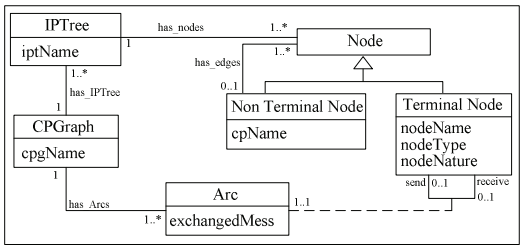

Fig. 11. VDS Concepts

More precisely an IPTree, which is a tree modelling an individual process, gathers terminal nodes and non terminal nodes. A non terminal node, which corresponds to a composite activity of VP2M, gathers nodes (*has_edges* relationship). On the other hand, a terminal node, which corresponds to either a version of atomic activity or a version of event of VP2M, has the following data structure:

- *nodeName*: name of the node that corresponds to the name of the corresponding version of atomic activity or event
- *nodeType*: type of the node that can be an activity or an event
- *nodeNature*: nature of the node that can be public or private for activities, or start, end or intermediate for events

In addition, these IPTrees may be linked by arcs between terminal nodes (*i.e.*, leaves) representing versions of public activities, thus indicating exchanged messages between the corresponding activities.

Table III gives the mapping rules from VP2M onto VDS. In addition we give a set of algorithms implementing these mapping rules. Due to lack of space, Fig. 12 focuses on the two main algorithms implementing the mapping from a version of collective process to its corresponding CPGraph (*MapOntoCPGraph*), and implementing the mapping from a version of individual process to its corresponding IPTree (*MapOntoIPTree*). Thus *MapOntoArc*, addressing the mapping from a version of interaction to an arc, *MapOntoTerminalNode*, addressing the mapping of a version of activity or version of event to a terminal node, and *MapOntoNonTerminalNode*, addressing the mapping of composite activity to a non-terminal node, are not presented in the paper.

TABLE III. MAPPING VP2M ONTO VIEW DATA STRUCTURE

VP2M Concepts	VDS Concepts
Version of Collective Process	CPGraph
Version of Individual Process	IPTree
Component	Node
Composite Activity	NonTerminalNode
Version of Atomic Activity	TerminalNode Type='activity'
Version of Event	TerminalNode Type='event'
Version of Interaction	Arc

The MapOntoCPGraph and MapOntoIPTree algorithms are given in Fig. 12.

```
Function MapOntoCPGraph (vcp : Version of Collective Process) : CPGraph
Local
   vip : Version of Individual Process
   vi : Version of Interaction
   vds_a : Arc
   vds_ipt : IPTree
   vds_cpg : CPGraph
Begin
    vds_cpg.cpgName = vcp.name // gives a name to the current CPGraph
    For Each vip In getVersionOfIndividualProcess(vcp) // from the VP2M
        vds_ipt = MapOntoIPTree(vip.start_with)  // builds a tree for partner vip
        Add_IPTree(vds_ipt,vds_cpg) // adds this tree to the current CPGraph
    End For
    For Each vi In getVersionOfInteraction(vcp) // from the VP2M
        vds_a = MapOntoArc(vi)  // builds an arc
        Add_Arc(vds_a,vds_cpg)  // adds this arc to the current CPGraph
    End For
    Return vds_cp
End Function

Function MapOntoIPTree (c : Component) : IPTree
Local
    comp : Component
    vds_tn : Terminal Node
    vds_ntn : Non Terminal Node
    vds_ipt : IPTree
Begin
    If IsVersionOfAtomicActivity(c) or IsVersionOfEvent(c) Then
        vds_tn = MapOntoTerminalNode(c) // build a terminal node
        Add_TN(vds_tn,vds_ipt) // add this node to the VDS
        Return vds_ipt
    Else
        vds_ntn = MapOntoNonTerminalNode(c) // build a non terminal node
        Add_NTN(vds_ntn,vds_ipt) // add this node to the VDS
        For Each comp In c.is_composed_of // for each of its components
            Return MapOntoIPTree (comp)
        End For
    End If
End Function
```

Fig. 12. MapOntoCPGraph and MapOntoIPTree Algorithms

These previous algorithms use the following set of functions for handling graphs and trees (VDS functions) and for handling versions of individual processes (VP2M functions):

- Add_IPTree(ipt,cpg): adds the IPTree *ipt* to the CPGraph *cpg*,
- Add_Arc(a,cpg): adds the Arc *a* to *cpg*,
- Add_TN(tn,ipt): adds the TerminalNode *tn* to *ipt*,
- Add_NTN(n,ipt): adds the Nonterminal Node *n* to *ipt*,
- IsVersionOfActivity(a): returns True if *a* is a version of activity, otherwise returns False,
- IsVersionOfEvent(e): returns true if *e* is a version of event, otherwise returns False,
- getVersionOfIndividualProcess(vcp): returns the set of versions of individual process involved in a version of collective process *vcp*,
- getVersionOfInteraction(vcp): returns the set of version of interaction for the version of collective process *vcp*.

Fig. 13 partially illustrates the mapping of the second version of the Subsea Pipeline process onto the VDS data structure. This version of the collective process (SP.2) involves two versions of individual processes (TPS-SP.2 and SAROST-SP.1), and several versions of interaction (only three of them are represented in Fig. 13). Arcs corresponding to these versions of interaction are visualised in blue.

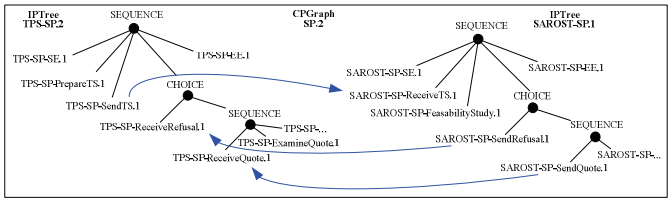

Fig. 13. CPGraph, IPTrees and Arcs for the Second Version of the Subsea Pipeline Process

2) Algorithms for Deducing Views. As explained before, the mapping step results in CPGraph composed of a set of trees and a set of arcs between terminal nodes of these trees, corresponding to versions of interaction. The recommended algorithms for views deduction have as input an initial CPGraph (containing all involved partners along with their public and private activities and events) and return as result a final CPGraph, consistent with the chosen view). These algorithms support activity extraction from IPTrees according to the requested view:

- public activities from IPTrees of a CPGraph for a collective view of an inter-organisational process modelled as a collective process in VP2M,
- public and private activities from a specific IPTree for a local view of an intra-organisational process modelled as an individual process in VP2M,
- public activities from a specific IPTree of a CPGraph along with public activities of its interacting IPTrees within the corresponding CPGraph for a global view of a partner in an inter-organisational process modelled as a collective process in VP2M,
- public and private activities from a specific IPTree of a CPGraph along with public activities of its interacting IPTrees within the corresponding CPGraph for a mixed view of a partner in an inter-organisational process.

We have provided in Fig. 14 the algorithm *Extract* to support activities extraction from IPTrees. Its scope parameter indicates the activities to be extracted: *Public* means that only public activities have to be extracted while *All* means that both public and private activities have to be extracted. Note that the *Reduce* function removes useless non-terminal nodes: for instance, if a non-terminal node is a sequence having only one child terminal node, the non-terminal node is replaced by its child terminal node (*e.g.*, sequence (a,b,sequence(c),d) is reduced to sequence (a,b,c,d)).

```
Function Extract (scope : string, vds_ipt : IPTree) : IPTree
Local
    vds_ipt_res : IPTree
    vds_n : Node
Begin
    For Each vds_n In vds_ipt.hasNodes
        If IsTN(vds_n) Then
            If vds_n.Type='Event' Then
                Add_TN(vds_n,vds_ipt_res)
            Else // vds_n is an activity
                If scope='All' or vds_n.Nature=scope Then
                    // This node has to be kept
                    Add_TN(vds_n,vds_ipt_res)
                End If
            End If
        Else // vds_n is a non terminal node
            Add_NTN(vds_n,vds_ipt_res)
        End If
    End For
    vds_ipt_res = Reduce(vds_ipt_res)
    Return vds_ipt_res
End
```

Fig. 14. Extract Algorithm

In addition, we have provided two algorithms for view deduction: the first one holds for collective view deduction while the second one holds for global and mixed view deduction. These algorithms are given in Fig. 15 and Fig. 16. Note that it is useless to deduce the local view of an intra-organisational process or of a specific partner involved in an inter-organisational process as this view is the corresponding IPTree.

```
Function CollectiveView (vds_cpg : CPGraph) : CPGraph
Local
    vds_ipt, vds_ipt_pub : IPTree
    vds_cpg_res : CPGraph
    vds_a : Arc
Begin
    For Each vds_ipt In vds_cpg.has_IPTree
        vds_ipt_pub = Extract('public',vds_ipt)  // extracts only public activities
        Add_IPTree(vds_ipt_pub,vds_cpg_res) // adds this tree
    End For
    For Each vds_a In vds_cp.has_Arcs  // we keep all the arcs
        Add_Arc(vds_a,vds_cpg_res)  // adds this arc
    End For
    Return vds_cpg_res
End Function
```

Fig. 15. CollectiveView Algorithms

More precisely, the CollectiveView algorithm presented in Fig. 15 returns only the public activities for all the involved IPTrees in a CP. The Extract function supports this selection. The GlobalMixedView algorithm presented in Fig. 16 holds

for both global view and mixed view deduction. To deduce the global view of a version of individual process within a version of collective process, the value of the scope parameter must be Public while to deduce its mixed view the scope parameter value must be All.

```
Function GlobalMixedView
      (vds_cpg : CPGraph, partnerName : string, scope : string) : CPgraph
Local
   vds_ipt, vds_ipt_pub : IPTree
   vds_setof_ipt : { IPTree }
   vds_cpg_res : CPGraph
   vds_a : Arc
Begin
    vds_ipt = Tree(partnerName)
   // returns the IPTree corresponding to partnerName
    vds_ipt_pub = Extract(scope,vds_ipt)
   // extracts activities that can be Public or both Public and Private (All)
    Add_IPTree(vds_ipt_pub,vds_cpg_res) // adds this tree
    vds_setof_ipt = InteractingPartners(vds_cp, partnerName)
    // returns the set of IPTree of partners interacting with partnerName within vds_cp
    For Each vds_ipt In vds_setof_ipt
         vds_ipt_pub = Extract('public',vds_ipt)  // extracts only public activities
         Add_IPTree(vds_ipt_pub,vds_cpg_res) // adds this tree
    End For
    For Each vds_a In vds_cp.hasArcs
         If not exists( Partners(vds_int) minus vds_setof_ipt) Then
              // Partners returns  the 2 partners involved in vds_a
              // exists & minus means that these 2 partners are in vds_setof_ipt
              Add_Arc(vds_a,vds_cpg_res)  // adds this arc
         End If
    End For
    Return vds_cp_res
End Function
```

Fig. 16. MixedView Algorithm

To illustrate these deductions, Fig. 17 and Fig. 18 give the CPGraphs corresponding respectively to the collective view and to the mixed view of TPS for the second version of the Subsea Pipeline process. In Fig. 17, only public activities are extracted; in Fig. 18, regarding TPS both public and private activities are extracted while regarding SAROST only public activities are extracted. Differences between these two views are highlighted in red: for instance, TPS-SP.PrepareTS.1, which is a version of a private activity of TPS, is only visualised in Fig. 18. Due to lack of space, we only give a partial representation of these graphs.

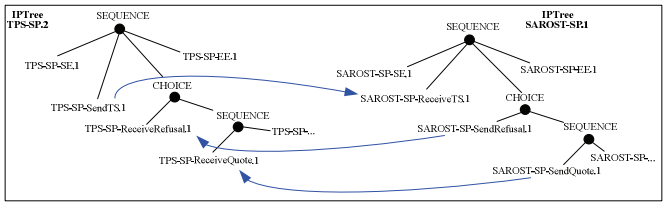

Fig. 17. Collective View for the second version of the Subsea Pipeline Process

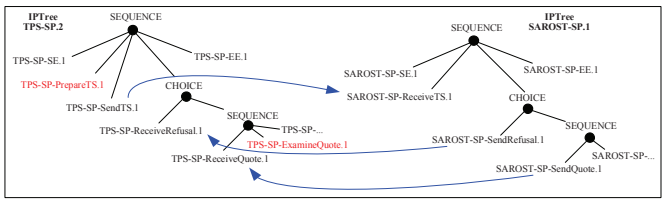

Fig. 18. Mixed View of TPS for the second version of the Subsea Pipeline Process

VI. Discussion and Conclusion

This paper has addressed loosely coupled inter-organisational process flexibility issue, which is an important challenge to address in the Business Process Management area. The approach advocated in this paper is the VP2M meta-model, gathering the core concepts of loosely coupled IoP and fulfilling the main requirements of [15]. VP2M also incorporates version modelling capability and keeps track of process, individual process, collective process, activity, operation, role, informational resource, event and interaction evolution, variability and adaptation. The paper has also addressed the dynamic aspects of IoPs version management, defining state charts for IoP versions and corresponding operations (create, update, delete, validate, derive and disable/enable operations) along with algorithms for deduction of views, which were introduced to address process abstraction and process assembly. Finally, the paper has illustrated the modelling of a flexible inter-organisational process, the Subsea Pipeline IoP.

Advantages of our contribution are the following. Firstly, VP2M is comprehensive as it integrates the main perspectives of processes and considers versioning of concepts related to all process perspectives. Moreover, as illustrated in the Subsea Pipeline IoP, VP2M supports the modelling of processes that may move from individual processes to collective process. Secondly, VP2M is a specific meta-model, independent from any language or notation, so we can generate executable and/or graphical specifications from instances of VP2M (as we did for instance in [33] for intra-organisational processes). Thirdly, the notion of version is well suited to address IoP flexibility issue and more precisely flexibility by variability, by evolution and by adaptation if adaptation can be defined at design-time. Fourthly and lastly, VP2M fulfils the main requirements of [15], and more particularly process *abstraction* and *process assembly*, using the notion of view and providing algorithms implementing their deduction. VP2M also fulfils the *modelling at the design level* and the *global business information schema* requirements thanks to, for this latter, versions of informational resources and versions of interactions.

Our contribution has three main drawbacks, which will be addressed in future works. Firstly, we started to improve our contribution in addressing not yet fulfilled requirements of [15]. More precisely, we have planned to introduce the notion of context for versions of IoPs in order to feature them and ease their reuse [34], providing a language for context specification and retrieval. Note that the notion of context also seeks to distinguish flexibility by evolution from flexibility by variability. Due to lack of space, we did not report our findings on this. We also have planned to implement our approach to define a collaborative modelling framework well suited to address intensive-knowledge IoP modelling, and to support IoP flexibility at run-time.

Secondly, we have foreseen to extend and implement view deduction algorithms. We have to extend them in order to take into account organisational and informational perspectives of process in IPTree. Thirdly, we have planned to evaluate the representational capability of our meta-model with the Bunge

Wand-Weber (BWW) ontology [35], which is used as a theoretical framework to evaluate expressiveness of information system analysis and design modelling languages, in particular to evaluate their expressiveness. We recently started this evaluation but have not made enough progress to report on it. We also have planned to evaluate VP2M usability, *i.e.*, VP2M acceptance by BPM practitioners.

VII. REFERENCES

[1] M. Dumas, M. La Rosa, J. Mendling, and H. Reijers, "Fundamentals of Business Process Management", *Springer*, 2013.

[2] M. Weske, "Business Process Management: Concepts, Languages, Architectures", *Springer*, 2007.

[3] M. Reichert, and B. Weber, "Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies", *Springer*, 2012.

[4] M. Rosemann, and W. van der Aalst, "A Configurable Reference Modeling Language", *Information Systems*, vol. 32, n°1, 2007, pp. 1–23.

[5] A. Hallerbach, T. Bauer, and M. Reichert, "Capturing Variability in Business Process Models: the Provop Approach", *Software Maintenance*, vol. 22, n°6-7, June 2010, pp. 519–546.

[6] M. Adams, A. ter Hofstede, D. Edmond, and W. van der Aalst, "Dynamic and Extensible Exception Handling for Worklows: A Service-Oriented Implementation", *Int. Conference on Cooperative Information Systems*, Vilamoura, Portugal, November 2007, pp. 95–112.

[7] S. Rinderle, M. Reichert, and P. Dadam, "Flexible Support of Team Processes by Adaptive Workflow Systems", *Distributed and Parallel Databases*, vol. 16, n°1, 2004, pp. 91–116.

[8] X. Zhao, and C. Liu, "Version Management for Business Process Schema Evolution", *Information Systems*, vol. 38, n°8, 2013, pp. 1046–1069.

[9] M. Adams, A. ter Hofstede, D. Edmond, and W. van der Aalst, "Worklets: a Service-Oriented Implementation of Dynamic Flexibility in Workflows", *Int. Conference on Cooperative Information Systems*, Montpellier, France, October 2006, pp. 291–308.

[10] E. Andonoff, L. Bouzguenda, and C. Hanachi, "Specifying Web Workflow Services for Finding Partners in the Context of Loose Inter-organizational Workflow", *Int. Conference on Business Process Management*, Nancy, France, September 2005, pp. 120–136.

[11] W. van der Aalst, "Loosely Coupled Interorganizational Workflows: Modeling and Analyzing Workflows crossing Organizational Boundaries", *Information and Management*, vol. 37, n°2, 2000, pp. 67–75.

[12] I. Ben Said, MA. Chaabane, R. Bouaziz, and E. Andonoff, "Flexibility of Collaborative Processes using Versions and Adaptation Patterns", *Int. Conference on Research Challenges in Information Science*, Athens, May 2015, pp. 440–451.

[13] I. Chebbi., S. Dustdar, and S. Tata, "The View-based Approach to Dynamic Inter-Organizational Workflow Cooperation", *Data and Knowledge Engineering*, vol. 56, n°2, 2006, pp. 139–173.

[14] D. Knuplesh, M. Reichert, J. Mangler, S. Rinderle-Ma, and W. Fdhila, "Towards Compliance of Cross-Organizational Process and their Changes: Research Challenges and State of Research", *Int. Workshop on Business Process Security*, within *Int. Conference on Business Process Management*, Tallinn, Estonia, September 2012, pp. 649–661.

[15] S. Lippe, U. Greiner, and A. Barros, "A Survey on State-of-the-Art to Facilitate Modeling of Cross-Organizational Business Processes", *Int Workshop on XML for Business Process Management*, Karlsruhe, Germany, March 2005, pp. 7–22.

[16] MA. Chaabane, E. Andonoff, R. Bouaziz, and L. Bouzguenda, "Versions to Address Business Process Flexibility Issue". *Int. Conference on Advances in Databases and Information Systems*, Riga, Latvia, September 2009, pp. 2–14.

[17] C. Ekanayake, M. La Rosa, A. ter Hofstede, and MC. Fauvet, "Fragment-based Version Management for Repositories of Business Process Models", *Int. Conference on Cooperative Information Systems*, Hersonissos, Crete, Greece, October 2011, pp. 20–37.

[18] M. Kradolfer, and A. Geppert, "Dynamic Workflow Schema Evolution based on Workflow Type Versioning and Workflow Migration", *Int. Conference on Cooperative Information Systems*, Edinburgh, Scotland, September 1999, pp. 104–114.

[19] M. Pesic, H. Schonenberg, and W. van der Aalst, "Constraint-based Workflow Models: Change made Easy", *Int. Conference on Cooperative Information Systems*, Vilamoura, Portugal, November 2007, pp. 77–94.

[20] D. Müller, M. Reichert, and J. Herbst, "A New Paradigm for the Enactment and Dynamic Adaptation for Data-driven Process Structures", *Int. Conference on Advanced Information Systems Engineering*, Montpellier, France, June 2008, pp. 48–63.

[21] W. van der Aalst, M. Weske, and D. Grünbaur, "Case Handling: a New Paradigm for Business Process Support", *Data and Knowledge Engineering,* vol. 53, n°2, 2005, pp.129–162.

[22] S. Nurcan, and MH. Edme, "Intention-driven Modeling for Flexible Workflow Applications", *Software Process: Improvement and Practice*, vol. 10, n°4, 2005, pp. 363–377.

[23] G. Bruno, F. Dengler, B. Jennings, R. Khalaf, S. Nurcan, M. Prilla, M., Sarini, M., Schmidt, R., and R. Silva, "Key Challenges for Enabling Agile BPM with Social Software", *Software Maintenance and Evolution: Research and Practice*, vol. 23, n°4, June 2011, pp. 297–326.

[24] F. Dalpiaz, E. Cardoso, G. Canobbio, P. Giorgini, and J. Mylopoulos, "Social Specifications of Business Processes with Azzurra", *Int. Conference on Research Challenges in Information Science*, Athens, Greece, May 2015, pp. 7–18.

[25] M. Döhring, B. Zimmermann, and L. Karg, "Flexible Workflows at Design-time and Run-time using BPMN2 Adaptation Patterns".*Int. Conference on Business Information Systems*, Poznan, Poland, June 2011, pp. 25–36.

[26] I. Ben Said, MA. Chaabane, E. Andonoff, and R. Bouaziz, "Extending BPMN 2.0 Meta-model for Process Version Modelling", *Int. Conference on Enterprise Information Systems*, Lisbon, Portugal, April 2014, pp. 384–393.

[27] W. Fdhila, S. Rinderle-Ma, and M. Reichert, "Change Propagation in Collaborative Processes Scenarios", *Int. Conference on Collaborative Computing: Networking, Applications and Worksharing*, Pittsburgh, Pennsylvania, USA, October 2012, pp. 452–461.

[28] W. Fdhila, C. Indiono, S. Rinderle-Ma, and M. Reichert, "Dealing with Change in Process Choreographies: Design and Implementation of Propagation Algorithms", *Information Systems*, vol. 49, 2015, pp. 1–24.

[29] S. Boukhedouma, M. Oussalah, Z. Alimazighi, and D. Tamzalit, "Adaptation Patterns for Service-based Inter-Organizational Workflows", *Int. Conference on Research Challenges in Information Systems*, Paris, France, May 2013, pp. 1–10.

[30] S. Boukhedouma, M. Oussalah, Z.Alimazighi, and D. Tamzalit, "Flexible Loosely Coupled Inter-Organizational Workflows using SOA", *Int. Conference on Computer Systems and Applications*, Ifrane, Morocco, May 2013, pp. 1–8.

[31] R. Eshuis, and P. Grefen, "Constructing Customized Process Views", *Data and Knowledge Engineering*, vol. 64, n°2, 2008, pp. 419–438.

[32] A. Tahamtan, and J. Eder, "View Driven Inter-Organisational Workflows", *Intelligent Information and Database Systems*, vol. 6, n°2, 2012, pp. 93–112.

[33] I. Ben Said, MA. Chaabane, and E. Andonoff, "A Model Driven Engineering Approach for Modelling Versions of Business Processes using BPMN", *Int. Conference on Business Information Systems*, Berlin, Germany, May 2010, pp. 254–267.

[34] M. Rosemann, R. Jan, and F. Christian, "Contextualization of Business Processes", *Business Process Integration and Management*, vol. 3, n°1, 2008, pp. 47–60.

[35] Y. Wand, and R. Weber, "An Ontological Analysis of some Fundamental Information System Concepts", *Int. Conference on Information Systems*, Minneapolis, Minnesota, USA, December 1988.