Extending Feature Models to Express Variability in Business Process Models

Riccardo Cognini, Flavio Corradini, Andrea Polini, and Barbara Re

Computer Science Division
University of Camerino, Camerino, Italy
{firstname.lastname}@unicam.it

Abstract. In complex organizations Business Processes tends to exist in different variants that typically share objectives and part of their structure. In recent years it has been recognized that the explicit modeling of variability can brings important benefits to organizations that can more easily reflect on their behavior and more efficiently structure their activities and processes. Particularly interesting in this respect is the situation of the Public Administration that delivers the same service using many different and replicated processes. The management of such complexity ask for methods explicitly supporting the modeling of variability aspects for Business Processes. In this paper we present a novel notation to describe variability of Business Processes and an approach to successively derive process variants. The notation takes inspiration from feature modeling approaches and has been implemented in a real tool using the ADOxx platform. The notation, and the corresponding approach, seems particularly suitable for the Public Administration context, and it has been actually experimented in a complex real scenario.

1 Introduction

In complex organizations Business Processes (BP) tends to exist in different variants that typically share objectives and part of their structure. In recent years it has been recognized that the explicit modeling of variability can brings important benefits to organizations that can more easily reflect on their behavior and more efficiently structure their work.

The delivery of services to citizens by Public Administrations (PAs) can certainly be re-conducted to such a situation. In this case the PA as a whole can be considered as a single organization in which the same BP could be declined in many different forms. So for instance, a residence move service will be supported by a BP that at a certain level of abstraction is described by a specific law also with reference to specific activities that have to be performed. Successively the possible many departments constituting the PA will independently implement their services, and supporting process, taking into account specific constraints related to the specific characteristic of the department itself. All this BP models share many characteristics but without a suitable support to represent such variability it will be difficult to share knowledge among the different part of the organization.

The case of PA is particularly interesting in reference to the possibility of representing variability for BP. In fact at a certain level of abstraction, and with respect to a specific process, all the departments will share the same abstract process. Nevertheless when detailed activities have to be introduced in order to make the service concrete the process models start to differentiate in order to include specific department characteristics [5]. For instance it is possible that in a big municipality different activities related to residence move will be carried on by different offices, while in a small municipality they will be carried on within the same office.

To solve this gap it is necessary to introduce a modeling approach that is able to represent law constraints and variability according different PA organizational structures.

To do that, we presents the Business Process Feature Model (BPFM) notation that combines in a new notation concepts coming both from feature modeling and from BP modeling. The notation permits to represent activities, their partial execution order, and involved data objects. A BPFM model collects all the possible BP variants, and via a configuration step it is possible derive the most suitable one for the specific organization. From a PA point of view, a BP manager configures the BPFM model according to the PA organizational structure. Then using a set of mapping rules we defined, BP manager can derive BP fragments. These fragments can be further enriched with control flow information considering specific characteristics of the PA. This two stages process to variant definition seems particularly suitable in a context in which all variability dimensions cannot be fully defined a priori. This is the case for instance of organizational aspects that can impact on the structure of a BP to be deployed, and for which variability aspects cannot be easily enumerated a priori.

The approach has been applied, with encouraging results, in the SUAP case study with reference to the *Start-up Certified Notification* scenario. The service refers to the activities that the Italian PAs have to put in place in order to permit to entrepreneurs to set up a new company or more in general to organize a business activity. SUAP includes more than 110 BPs, that are different considering the request target, nevertheless all of them are quite similar and overall they could be considered a single process family. Using the ADOxx development platform we also implemented a modeling environment supporting the usage of the BPFM notation.

The paper is organized as follow. Section 2 reports some background material, while Section 3 reports relevant related works. Successively, Section 4 gives an overview of the approach, and Section 5 shows the proposed notation. Section 6 presents the developed tool. Validation activities are discussed in Section 7. Finally Section 8 reports conclusions and opportunities for further research.

2 Feature Modeling

Feature modeling is an approach emerged in the context of Software Product Lines to support the development of a variety of products from a common plat-

form. The approach aims at lowering both production costs and time in the development of individual products sharing an overall reference model, while allowing them to differ with respect to specific scenarios to serve, e.g. different markets [15]. In the last years feature modelling have been used also to represent commonality and variability in *Business Information Systems*, introducing the concept of family of BP.

A FM is a graphical model that, using a tree representation where the root represents the general product to develop, permits to express different relationships among the possible features that can be included in a specific variant of the product. In particular, in the first feature modeling approach proposed, named Feature-oriented Domain Analysis (FODA), *mandatory*, *optional* or *alternative* constraints on features have been introduced [9]. A *Mandatory* feature represent a characteristic that each product variant must have. For instance considering the production of different mobile device types we could define a constraint requiring that any mobile device variant have to include a screen. An *Optional* feature is used to represent characteristics that a product can have but a fully functional product can also be derived without including such a feature. For instance this could be the case of mechanisms supporting connection to 4G networks that could be included only in high-profile products. An *Alternative* feature represents characteristics that cannot be present together in a product. For instance a mobile device can have a standard screen or a touch screen, but not both. Researchers have proven that basic FM models are too restrictive to represent all the relationships between features which are useful to characterize a family of products [1]. As a result the FM notation has been extended to permit the definition of feature cardinality, permitting to define how many features in a set are needed to have a working product. It is possible then to express relationships such as *"at least one feature in a set of features is needed in each product"*. This is done via *OR features* constraints. Additionally, *include relationship* constraints are used to express that a feature selection implies the selection of another feature that is on a different part of the tree, and *exclude relationship* constraints are used to express that a feature selection requires to discard another one that is on a different part of the tree.

Once a feature model has been defined it is possible to derive a specific product defining a *configuration* that express explicit features selection, and according to the constraints defined in the feature model.

3 Related Works

BP modeling has been identified as a fundamental phase in order to better understand how to behave and organize activities within a complex organization. Different classes of languages to express BP models have been proposed in the last years such as BPMN 2.0 [12], EPC [18] or YAWL [20]. These notations permits to specify BPs even if they do not have mechanisms to represent classes of similar BP that can be represented as a family. Indeed this has emerged as an important characteristics since in similar contexts processes can share several

characteristics which are difficult to reuse with standard notation. As a result in recent years the interest towards techniques for modeling such variability has clearly raised [2].

Modeling variable BP is the ability to represent in a single model many alternative BPs sharing the same goal [16]. In order to describe variable BPs several approaches have been proposed, in some case extending already available notations. Relevant examples are certainly languages such as C-EPC [17], Configurable integrated EPC (C-iEPC) [11], vBPMN [4] or C-YAWL [7]. Also language independent approaches have been proposed. Among the others PROVOP [8] and PESOA [19] are probably the most used.

Differently from our proposal such modeling languages permits to derive variants for which the control flow is fully determined. The configurable model includes all the possible control flow relations and a subset of them are included in a derived variant. This approach cannot be applicable when the characteristics to consider to derive the variant are not enumerable. Our approach instead permits to derive variants for which the control flow have to be successively refined considering information available only at configuration time.

Alternative approaches are those based on the declarative paradigm such as CMMN [13] and Declare [14]. Nevertheless differently from our proposal such approaches do not intend to provide variants with a fully specified control flow and typically defer the definition of a precise order between the activities till their execution.

4 Overview of the Approach

The proposed approach is organized in four main steps (Fig. 1) and it results to be particularly suitable in situations in which an abstract definition of a process needs to be successively refined to consider specific aspects of the deployment context, such as the specific characteristics of the organization supporting the process itself. This is a quite common situation for processes supporting PA services to citizens. In such a case objectives and activities constituting the process are general and independent from the specific characteristics of the offices delivering the service itself. Nevertheless the precise definition of the process, in terms of roles and ordering of the activities, depends from deployment related aspects such as for instance the organizational model.

Input of the proposed approach are the laws regulating the provisioning of a service, while the final output will be a BP variant that can be deployed according to the characteristics of the service under analysis, and the organizational model of the Public Administration which delivers the service to the citizen. In particular the approach is organized in 4 successive steps:

– The first step aims at defining a general model that can be successively constitute the basis for the definition of a process variant for the specific deployment context. The model will be codified using the BPFM notation presented in the

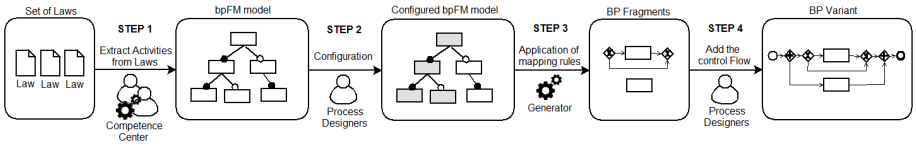

Fig. 1. Steps of the Approach.

next section. This step include knowledge acquisition through the study of legal and regulatory frameworks governing the delivery of the PA service under study. This step should be carried on only once for each service delivered by the PAs. The activity performed by a focus group or a competence centre, will permit to derive a model that will include only the activities that have to be carried on, the relations among them, and the data structure they possibly get in input or produce in output (as said this information are codified in a BPFM model as illustrated in the following).

- The second step foresees the refinement of the previously defined model taking into account the specific needs of the service that has to be delivered and of its deployment context. Similarly to what it is done in feature modeling this step foresees the definition of a *configuration* on the BPFM model which will permit to define a specific variant from the BP family.
- The third step takes in input activities and data objects resulting from the configuration defined in the previous step. Through the application of mapping rules we define it is possible then to automatically derive BP fragments representing portions of the behaviour that has to be completed to reach the goal of the service to be delivered.
- The last step concerns the derivation of the fully specified BP variant starting from the generated BP fragments. At this stage process designers add control flow relationships among the generated BP fragments, also taking into account the specific characteristics of the PA organization that needs to deliver the service to citizens. It is worth mentioning that the same activity could be associated to different roles in different BP variants, as a result of possible different organizational models for different PA offices.

For the sake of space in this paper we mainly focus on the notation we introduced to perform the first step described above, and we will not report the mapping rules needed to perform step 3 that can be retrieved here [3].

5 Modeling Variability with BPFM

The BPFM notation intends to provide a tool to model a family of BPs that is identified by the root element of a model. In a BPFM model feature elements represent activities that can be included or not in a BP variant successively derived. Activities are decomposed going up-to-down in a tree model giving the opportunity to introduce variability aspects thanks the possibility of using a superset of the connectors used in the standard FM notation (see Fig. 2).

Activities can be atomic in case they are leafs of the BPFM tree, or composed in case they are parents of other activities. Data objects are also included in BPFM permitting to include information that will be successively helpful for the definition of correct BP variants.

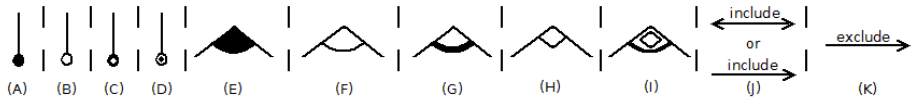

Fig. 2. BPFM Constraints.

Feature constraints express if an activity must or can be inserted in a BP variant, and if it must or can be included within any execution path at the instance level (i.e. real execution of the process). Feature (activity) constraints can be 1-to-1 or 1-to-n depending on how many features they refer to. Moreover each feature can be involved in many binary relations playing the role of the parent. Nevertheless an activity can also be the parent in just one 1-to-n relation.

Feature constraints can be binary or multiple depending on how many child activities are connected to a parent activity. With respect to the binary constraints we consider the following. A *Mandatory Constraint* requires that the connected child activity must be inserted in each BP variant, and it has also to be included in all execution paths (Fig. 2-A). A *Optional Constraint* requires that the connected child activity can be inserted (or not) in each BP variant and it could be included (or not) in each execution path (Fig. 2-B). A *Domain Constraint* requires that the connected child activity must be inserted in each BP variant but it could be included (or not) in each execution path (Fig. 2-C). A *Special Case Constraint* requires that the connected child activity can be inserted (or not) in each BP variant. When it is inserted it has to be included in each execution path (Fig. 2-D).

With respect to multiple constraints we consider the following. An *Inclusive Constraint* requires that at least one of the connected child activities must be inserted in each BP variant, and at least one of them have to be included in each execution path (Fig. 2-E). A *One Optional Constraint* requires that exactly one of the connected child activities has to be inserted in each BP variant, and it could be included (or not) in each execution path (Fig. 2-F). A *One Selection Constraint* requires that exactly one of the connected child activities has to be inserted in each BP variant, and it has to be included in each execution path (Fig. 2-G). A *XOR Constraint* requires that all the connected child activities must be inserted in each BP variant, and exactly one of them has to be included in each execution path (Fig. 2-H). A *XOR Selection Constraint* requires that at least one of the connected child activities has to be inserted in each BP variant, and exactly one of them has to be included in each execution path (Fig. 2-I). Finally, *Include* and *Exclude* relationships between activities are also considered according to the base definition of FM (Fig. 2-J and Fig. 2-K).

In BPFM the modeling of Data Objects plays also an important role. BPFM includes all types of BPMN 2.0 Data Objects and it uses the same symbols (Fig. 3-A). As well as in BPMN 2.0 Data Object elements can be connected as inputs and outputs to activities (features). In particular child features inherit Data Objects from the parent node, and if a Data Object is connected as input (or output) to a feature, all the child activities will need such Data Object.It is worth noting that given that different correct configurations could include or not some node, it is possible that different BP variants will include different sets of Data Objects.

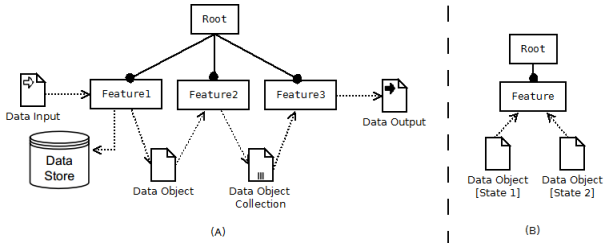

Fig. 3. Data Object in BPFM.

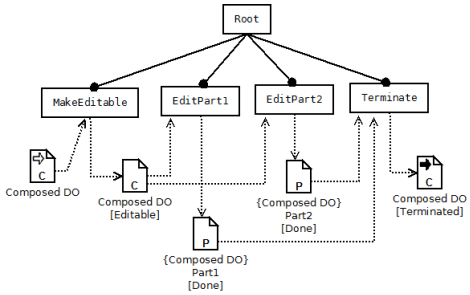

Fig. 4. Composed Data Object in BPFM.

In BPFM a status can be associated to a data. An activity can require or can generate a Data Object in a specific state and consequently can change its state. If the state is not explicitly reported the activity is state independent. A Data Object cannot be in two different states at the same time (Fig. 3-B). The state of a Data Object is represented with square brackets under the Data Object name. Moreoverin BPFM, differently from BPMN 2.0, we introduced the possibility to represent composite and *part-of* Data Objects that can be extended for each type of BPMN 2.0 Data Object (see for example Fig. 4).

- A composed Data Object indicates that the Data Object is composed by a set of specific block of data, and it is marked with the letter *C*.
- *Part-of* Data Object indicates that the Data Object is contained in a specific block of data, and it is marked with the letter *P*. It also explicitly refers to the Data Object of which it is part reporting the name of it inside curly brackets.

The increased expressiveness of BPFM in modeling data related information results to be particularly useful in modeling processes within the PA context where complex data structures and relations (forms, documents etc.) typically drive the execution of a BP.

Once a BPFM model have been derived a variant can be obtained thanks to the definition of a configuration. A configuration selects some of the features according to the constraints included in the model. This step is absolutely similar to what it is done with traditional FM models. Nevertheless thanks to the mapping defined in [3] a set of BP fragments will be immediately derived from a configuration. Successively fragments have to be composed by the modeler to finally derive a fully functional BP variant for the specific PA organization.

6 BPFM ADOxx Prototype

The modeling approach illustrated in this article is supported by a modeling environment that can be freely downloaded at the BPFM web page (http://www.omilab.org/web/bpfm). It has been developed thanks to the functionality made available by the ADOxx platform[1]. ADOxx is a set of tools developed in order to make easy the implementation of modeling environments based on meta-models [6] [10].

To derive a modelling environment we designed the BPFM meta-model according to what it is shown in Figure 5). Therefore *Activity* represents atomic or composed tasks. *Constraint* expresses the relationships between activities. Constraints can be *Binary Constraint* or *Multiple Constraint*. Then *Binary Constraint* is further specialized in four sub-classes that are *Mandatory*, *Optional*, *Domain* and *Special Case*. *Multiple Constraint* is specialized in five sub-classes that are *XOR Selection*, *XOR*, *Inclusive*, *Alternative* and *One Optional*. *Data Object* introduces input output data for activities they can be specialized in three sub-classes, they are *Data Input*, *Data Output* and *Data Store*. *Data Object Connector* representing the relationships between activity and Data Object that can be *Input Data Object Connector* or *Output Data Object Connector*.

Focusing on the *Activitiy*, they can be specified using the attribute *type* that can assume the following values: standard, service, send, receive, manual, user, script or business rules. Relationship between an *Activity* and *Constraint*, and vice-versa, are exclusively characterized as binary or multiple. For what concern constraints each activity can take in input zero or one binary constraint or one multiple constraint. There is one special activity, named root, that has zero input constraints. In output the activity can have zero or more binary constraint or one multiple constraint. Relationship between activities can also be expressed via *Include* or *Exclude* relationship. Each *Activity* can include/exclude zero or more *Activities*. From the other side, each *Activity* can be included/excluded by zero or more *Activity*.

[1] http://www.adoxx.org

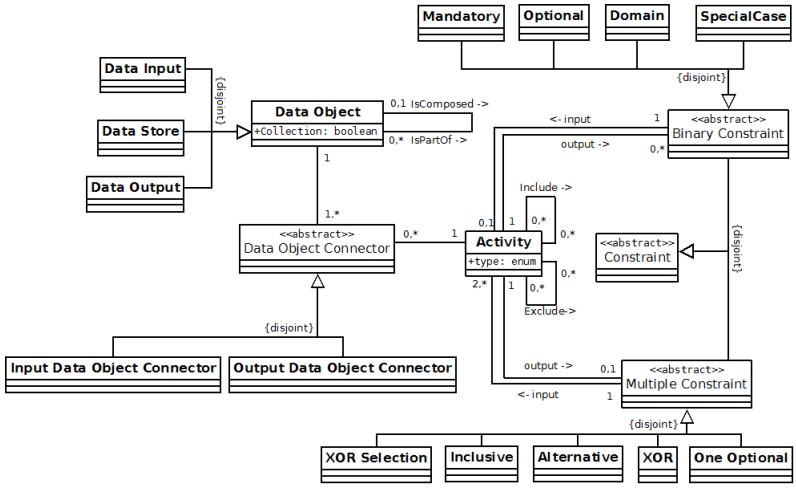

Fig. 5. BPFM Meta-Model

Regarding *Data Object* the attribute *Collection* specifies if the *Data Object* is a collection or not. *Data Object* has a self-relationship to represent the notion of composition. Each *Data Object* can be part of zero or one Data Object. On the other side each *Data Object* can be composed by zero or more parts. Focusing on the *Data Object* relationship with *Data Object Connector*, each *Data Object* must be connected to at least one *Data Object Connector*. For each *Data Object Connector* there is just one connected *Data Object*. Finally, *Data Object Connector* must be connected to an *Activity*, and an *Activity* can be in relationship to zero or more *Data Object Connector*.

Then according to the described meta-model BPFM ADOxx prototype has been developed. We first created all the elements, constraints and graphical representations discussed in Section 5 and then we include them in the BPFM model-type. Therefore using the resulting Modeling Toolkit, it is possible then to define BPFM models using a graphical editor

7 The SUAP case: Start-up Certified Notification

The described approach has been applied to model processes related to the Italian "Sportello Unico per le attività produttive" (SUAP). This is a service that the Italian Public Administrations have to put in place in order to permit to entrepreneurs to set up a new company. Among the many processes composing the service we refer here to the *Start-up Certified Notification* (SCIA). From the point of view of entrepreneurs this is just a notification. Instead if the point of view of the PA is considered, this is a quite complex process that ask to check the correctness and good faith of the application, mainly composed by self-certifications. Therefore it requires to involve, when needed, all the appointed offices in the same or different Public Administrations.

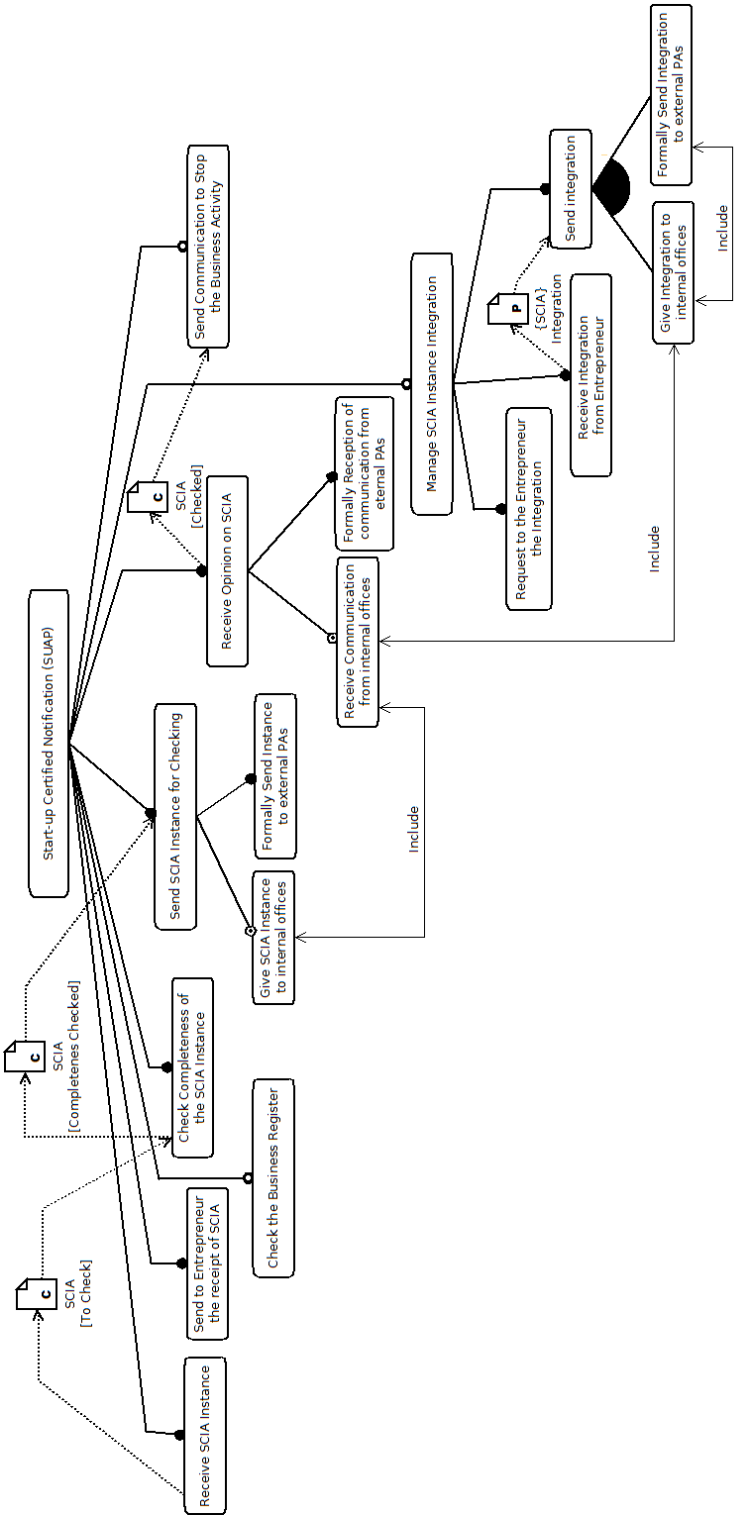

Fig. 6. Start-up Certified Notification BPFM model in ADOxx

Starting from the law BPFM can be generated representing the SCIA service (Fig. 6). This is a quite simple BPFM that at the same time seems sufficiently complex to show the potentialities of the notation. The root of the BPFM model represents the SCIA service, and as it can be observed also Data Objects are included. They are input/output for activities in the SCIA BP. For the sake of space we discuss here some interesting detail about the model.

Receive SCIA Instance activity is connected using a *Mandatory Constraint*, it is available in any SCIA variant as well as in any execution path since in any configuration it will obviously necessary to receive the application from the entrepreneur. The entrepreneur self-certification is sent to the PA offices and third parties administrations involved in the verification activity. They check the correctness of the self-certifications and give back feedback in order to clarify if the self-certifications are valid or not. These activities are represented by *Give SCIA instance to internal office* and *Formally send instance to external PAs* connected to *Send SCIA Instance to other PAs* via a *Special Case Constraint* and a *Mandatory Constraint* respectively. *Manage SCIA Instance Integration* is connected to the root using two *Domain Constraints*, so they have to be available in each BP variant and it is not always available in each execution path. Integration is asked to the entrepreneur to complete the self-declaration. *Send Communication to stop the Business Activities* is connected using two *Domain Constraints*, so they have to be available in each BP variant and it is not always available in each execution path. It could be that incomplete self-certification and some legal issues observed during check asks for the termination of the business activity.

Notwithstanding the complexity of the process modeling of the scenario has revealed that the notation permits to focus at different stage to different aspects. In particular the derivation of the BPFM model asks to the modeler to mainly focus on the function and data perspective, while the behavioral perspective is considered in step 4. This separation of concerns results to be particularly fruitful when complex scenario are considered.

8 Conclusion and Future Work

In this paper we presented a notation and a modeling environment to represent variability in business processes. The approach seems particularly suitable to derive process variants for services delivered by the PA. The first experiments made with the notation provided encouraging results and permitted to model quite easily a complex scenario and to derive the corresponding processes.

In the future we plan to continue the experimental work and to continue the implementation of the tool to support all the steps foreseen by the approach. Another important aspect we plan to investigate refers to the definition and introduction of mechanisms to verify that derived BP variants are valid with respect to the BPFM model constraints.

References

1. R. Capilla, J. Bosch, and K. C. Kang, editors. *Systems and Software Variability Management, Concepts, Tools and Experiences*. Springer, 2013.
2. R. Cognini, F. Corradini, S. Gnesi, A. Polini, and B. Re. Research challenges in business process adaptability. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pages 1049–1054. ACM, 2014.
3. R. Cognini, F. Corradini, A. Polini, and B. Re. Using data-object flow relations to derive control flow variants in configurable business processes. In F. Fournier and J. Mendling, editors, *BPM 2014 Workshops, LNBIP 202*, pages 1 – 12, 2015.
4. M. Döhring and B. Zimmermann. vbpmn: event-aware workflow variants by weaving bpmn2 and business rules. In *Enterprise, Business-Process and Information Systems Modeling*, pages 332–341. Springer, 2011.
5. E. Erkoçak and Ş. N. Açıkalın. Complexity theory in public administration and metagovernance. In *Chaos, Complexity and Leadership 2013*, pages 73–84. Springer, 2015.
6. H.-G. Fill and D. Karagiannis. On the conceptualisation of modelling methods using the adoxx meta modelling platform. *Enterprise Modelling and Information Systems Architectures-An International Journal*, 8(1), 2013.
7. F. Gottschalk, W. M. Van Der Aalst, M. H. Jansen-Vullers, and M. La Rosa. Configurable workflow models. *International Journal of Cooperative Information Systems*, 17(02):177–221, 2008.
8. A. Hallerbach, T. Bauer, and M. Reichert. Capturing variability in business process models: the provop approach. *Journal of Software Maintenance and Evolution: Research and Practice*, 22(6-7):519–546, 2010.
9. K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson. Feature-oriented domain analysis feasibility study. Technical report, DTIC Document, 1990.
10. H. Kühn. The ADOxx® Metamodelling Platform. In *Workshop on Methods as Plug-Ins for Meta-Modelling, Klagenfurt, Austria*, 2010.
11. M. La Rosa, M. Dumas, A. H. ter Hofstede, J. Mendling, and F. Gottschalk. Beyond control-flow: Extending business process configuration to roles and objects. In *Conceptual Modeling-ER 2008*, pages 199–215. Springer, 2008.
12. OMG. Business process model and notation version 2.0. Technical report, 2011.
13. OMG. Case Management Model and Notation, Version 1.0, May 2014.
14. M. Pesic, H. Schonenberg, and W. M. van der Aalst. Declare: Full support for loosely-structured processes. In *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*, pages 287–287. IEEE, 2007.
15. K. Pohl, G. Böckle, and F. J. v. d. Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag, 2005.
16. M. Reichert and B. Weber. *Enabling flexibility in process-aware information systems: challenges, methods, technologies*. Springer Science & Business Media, 2012.
17. M. Rosemann and W. M. van der Aalst. A configurable reference modelling language. *Information Systems*, 32(1):1–23, 2007.
18. A.-W. Scheer, O. Thomas, and O. Adam. Process modeling using event-driven process chains. *Process-Aware Information Systems*, pages 119–146, 2005.
19. A. Schnieders and F. Puhlmann. Variability mechanisms in e-business process families. In W. Abramowicz and H. C. Mayr, editors, *BIS*, volume 85 of *LNI*, pages 583–601. GI, 2006.
20. W. M. Van der Aalst and A. H. Ter Hofstede. Yawl: yet another workflow language. *Information systems*, 30(4):245–275, 2005.