Configuring the Variability of Business Process Models Using Non-Functional Requirements

Emanuel Santos[1,2], João Pimentel[1], Jaelson Castro[1], Juan Sánchez[2], and Oscar Pastor[2]

[1] Universidade Federal de Pernambuco, Centro de Informatica, Cidade Universitaria S/N, 50741-000 Recife, Brazil
{ebs,jhcp,jbc}@cin.ufpe.br

[2] Universidad Politecnica de Valencia, Camino de Vera, S/N, 46022. Valencia, Spain
{jsanchez,opastor}@dsic.upv.es

Abstract. The existence of variations in the organizational environment makes the configuration of business process models a complex activity, even for experienced business analysts. The increasing adoption of business processes models by software engineers as a input for requirements analysis strengthens the importance of adressing this issue. The challenge is to configure business processes to fit the organization better. We propose an approach that combines variability analysis and non-functional requirements to drive the configuration of a business process. Applying this approach we can analyze variability in the model in order to assess the impact of the choices on the process quality constraints - the non-functional requirements. Moreover, it provides a rationale for the selection of a specific configuration.

Keywords: Business Process Models, Business Process Configuration, Variability, Non-Functional Requirements.

1 Introduction

With the increasing interest of the software engineering community in using business process models as a source of requirements, raised the importance of representing variability on these models. Variability, on business process models, consists of defining alternative paths of execution in a workflow [1]. In this way, the process can be personalized for a specific context, e.g., for a foreign subsidiary of a corporation.

There are several approaches for representing variability in a business process model, like Schnieders and Puhlmann [2], Montero et al. [3] and La Rosa et al. [4]. However, the problem of choosing the most suitable alternative - the so-called process configuration - is not solved yet. In the industry, the configuration still is performed in an ad hoc basis, guided only by the analyst's experience. Some techniques have been proposed in academia, like the usage of questionnaires [4] and domain analysis [5], but these techniques are more concerned with the elicitation of variability than with the configuration itself.

I. Bider et al. (Eds.): BPMDS 2010 and EMMSAD 2010, LNBIP 50, pp. 274–286, 2010.

In this paper we propose an approach for performing business process configuration based on its non-functional requirements (NFR). These requirements, sometimes called quality requirements, define constraints that the process must comply to. We believe that non-functional requirements are a suitable criterion for guiding the process configuration, since they represent the high-level characteristics from which processes are usually evaluated - cost, performance, accuracy, and so on. Also, the solid foundations on which software systems NFR is built [6] provide plenty of techniques that can be borrowed and used in this new domain. Some recent works are already heading toward the integration of NFR and business process models [7] [8] [9].

We are going to present our approach using the Business Process Model and Notation - BPMN [10], since it is a well known and acknowledged notation in the software engineering community. However, this approach can be applied to any other process notation in which variability can be expressed.

In summary, the main contributions of this paper are twofold:

- The definition of an approach for business process configuration using non-functional requirements (NFR) as the selection criteria. We describe how to model variability in the business process (first phase), link the process variants to the NFR and use the linkages to select the best configuration for a selected NFR (second phase).
- The integration of current NFR techniques and algorithms, aiming to enable the automatic configuration of a business process. In this way, the configuration could be performed at design time, by a process analyst, or at run-time, by the system itself.

In Section 2 we are going to introduce the background of our research, namely BPMN and non-functional requirements. Following, we present the approach itself, in Section 3. The application of our approach is exemplified in Section 4. Related works are presented in Section 5. Finally, in Section 6, the conclusions are discussed.

2 Background

Business Process Model and Notation (BPMN) is a notation to model business processes in terms of their activities and supporting information [11]. The BPMN is based on the representation of activities flows and allow to represent different levels of details for different purposes. Figure 1 depicts the most commonly used BPMN elements and their graphical notation. In BPMN, the roles that participate in a process are represented by Pools and Lanes. Pools represent organizations and Lanes represent the participants or subdivisions of an organization. The process is composed by sub-processes and tasks, connected through flows of communication. There are elements that represent the events that start the process, that finish the process or that happen during the execution of process (i.e., intermediary events). With these elements the business analyst can represent, analyze and propose improvements to the business processes of an organization.

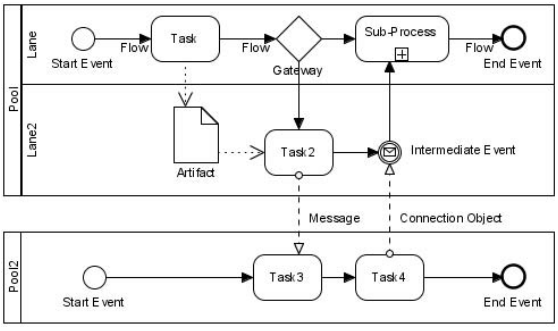

Fig. 1. BPMN elements

The non-functional requirements (NFR) are requirements that specify criteria to judge the operation of a software, rather than a specific behavior. It usually is represented in terms of qualities, or constraints, that the software should be concerned with. The NFR can be seen as properties observable at run-time - such as security or usability, or as properties embodied in the product - such as maintainability or scalability.

Chung et al. [6] describes a framework to model and analyze non-functional requirements. The NFR Framework is based on the concept of Softgoal that is a representation of non-functional requirements. The softgoals are characterized as the goals that the system should achieve but that have no clear achievement criteria. The NFR Framework uses the Softgoal Interdependency Graph (SIG) to refine and analyze the interation among softgoals using the decomposition and contribution analysis. The models generated using the NFR Framework can be grouped in reusable catalogues.

3 The Approach

Our approach aims to provide a way for configuring business process models, maintaining the rationale behind the selection of a specific process instance. It is divided in two phases, each one with two steps. The first phase consists of analyzing the business process model, which we are modeling with BPMN, in order to discover and represent possible variations of the model. In the second phase we analyze the non-functional requirements and perform the configuration itself.

We describe the variations in terms of variants and variation points [12]. The variation points are the subject of variation, in the case o BPMN could be tasks, events, artifacts, or pools. They are points where new elements can be added, replaced or removed to represent different behaviors of process. In its turn the variants describe the object of variation, for example, if a task could be performed in different ways each one could be represented as a variant.

3.1 Phase 1 - Elicitation and Representation of Variability in a BPMN Model

In this first phase we are going to elicit variability (Step 1.1) and represent it on the process model (Step 1.2), assuming that this have not been done yet. So, in this phase the aim is to identify and to organize the variations that can be found on a given business process.

Step 1.1 - Elicit variability. The elicitation of variability is the activity of identifying and discovering possible variations in a model. The goal is to identify different ways to carry on a process, what could result in the inclusion, changing or exclusion of elements on the model. To perform this elicitation it we use an information analysis framework [13] that explores different facets of the information and obtain new data about it. In the context of BPMN models we will use this framework to inquire the tasks, activities and sub-processes of model and identify new information about them. The use of this framework is as simple as making questions like Who? How? When?, which is very usual in requirements engineering.

The facets that can be identified, with the respective questions, are:

- Agentive (Who will perform the task?)
- Dative (Who will be affected by the task?)
- Objective (What are the objects consumed or produced by the task?)
- Extent (What are the degree of the task will be performed?)
- Process (How the action will be executed?)
- Conditional (In what conditions the task will be performed?)
- Locational (Where the task will be performed?)
- Temporal (When the task will be performed?)

Asking those questions to each element of the process model, we can identify a comprehensive set of possible variations on a business process. In the next step we are going to represent these variations in terms of the BPMN notation.

Step 1.2 - Describe variability. The elicitation results in a list of variations that need to be represented in order to reflect the nature of business process. So, in this step we put the variations in terms of the BPMN notation. In doing so we can apply these variations in the BPMN model while maintaining the consistence of the notation. As explained before, we represent the variations using the concepts of variation points and variants. Variation point is the place where the variation occurs, and each possible alternative for a variation point.

To describe the variants we are using an identifier, the point where should be inserted, the dependencies that can be present and a pattern of insertion. The patterns of insertion are already described in the literature [14]. They can be the insertion of sequences, parallelism, optional behavior, and so on. We identified that this set of patterns were too limited for this approach, so we complement them with patterns for deletion, insertion of lanes and substitution. The deletion pattern covers the case of a negative dependency that happens when a variant

excludes another variant. In this pattern all elements in the variation point are deleted and the beginning and the end of the variation point are linked directly. The insertion of a lane is a pattern specific for BPMN and is related to the inclusion of new roles in the model.

Variation points are described with an identifier (name), a type (task, link, sequence), a point of reference (begin and end), and a list of the variants that can be placed in it. Throughout the example of Section 4 we present some examples of variation points and variants.

3.2 Phase 2 - Analysis and Configuration

The variation points and the variants are essential inputs for performing the process configuration. However, this information by itself is not sufficient to identify which subset of variants results on the best process. So, in this phase we are going to link the variants to non-functional requirements and analyze which process configuration maximize a selected criterion.

Step 2.1 - Link variants to NFR. In this step the non-functional requirements (NFR) will be linked to the business process variants identified earlier. To do so, we first need to identify which NFR will be taken into consideration. This can be done interviewing people involved in the business process [9], using requirements catalogs [7] or with a mix of elicitation techniques.

Once the NFR are identified, we will perform the linking between the process variants and the requirements. These links will be represented using matrices, which is a usual and scalable solution for representing this kind of information. Moreover, matrices allow the building of views containing only a partial representation of the variants and the requirements, simplifying its analysis.

In Table 4 (Section 4) we provide a example of the variants to NFR matrix. The lines of the matrix are the process variants, grouped by its variation points, and the columns are the non-functional requirements. For each variant, we are going to define the impact of that variant on each NFR, in a scale of -1 to 1 (inclusive). A negative value in this scale means a negative impact, as well as a positive value means a positive impact. Zero is the neutral value, meaning that variant does not impact the NFR at all.

In order to make it more user friendly, this scale can be replaced by any other scale, provided that the required transformation is performed. For instance, let us consider the qualitative scale of the NFR Framework [6]. In the NFR Framework, the most positive impact on a non-functional requirement is *Make*, a partial positive impact is *Help*, a partial negative impact is *Hurt* and the most negative impact is *Break*. These values can be mapped, respectively, to 1, 0.5, -0.5 and -1, in our scale.

Now that we have the linkages between the process variants and the NFR, we can perform the configuration itself, in the next step.

Step 2.2 Perform the configuration. At this point we know the variation points and the variants of the business process, and how they impact the

non-functional requirements. Now we will use this data to support the configuration itself.

There are two possible ways for analyzing the impact of each configuration on the NFR: top-down analysis and bottom-up analysis. In the top-down analysis we select which non-functional requirement has the maximum priority, and then derive a process configuration that maximizes the selected NFR. Alternatively, in the bottom-up analysis we define a process configuration, by selecting a subset of variants, and then observe how this configuration affects the non-functional requirements.

These analysis can be performed semi-automatically. The algorithms to perform the evaluation of alternatives using non-functional requirements are already available in the literature [6] [15]. The choice of matrices as data structure allows the usage of even more sophisticated algorithms, in order to resolve dependencies and conflicts that may arise. However, it is up to the analyst to select the NFR used as criteria - in the top-down analysis - or the configuration that will be evaluated - in the bottom-up analysis.

Following, we present the analysis themselves:

Top-Down Analysis. The top-down analysis consists of obtaining an instance of the model based on the selection of a non-functional requirement. So, the analyst will define which NFR will be prioritized. Each variation point is evaluated to identify the variant that better fits the selected non-functional requirement. I.e., the variant which has the biggest positive impact on that NFR. This evaluation can be performed automatically. However, dependencies between variants have to be taken into consideration as well. If a variant X require the variant Y, the calculation will be performed considering X and Y altogether.

Bottom-Up Analysis. The bottom-up analysis consists of selecting a subset of variants and using the linkage matrix to calculate the impact of that configuration on the non-functional requirements. This way an analyst could, for instance, evaluate if the current configuration is satisfactory.

A good way of performing the configuration is to perform a top-down analysis and then evaluate subtle changes of the configuration, using bottom-up analysis. This way the analyst will have a starting point for the configuration and will be able to understand how his changes affect the non-functional requirements. At the end, the analyst will have not only the process instance, but also the rationale for choosing that instance. E.g., "this configuration maximizes the accuracy, while maintaining a low cost".

4 Running Example

In order to demonstrate the application of our approach, we will introduce an example of Conference Management. During the organization of a conference several activities are realized: the call for papers, the revision of papers, the organization of proceedings, and so on. The diagram on Figure 2 presents an excerpt of a process of revision and notification of acceptance in a small conference.

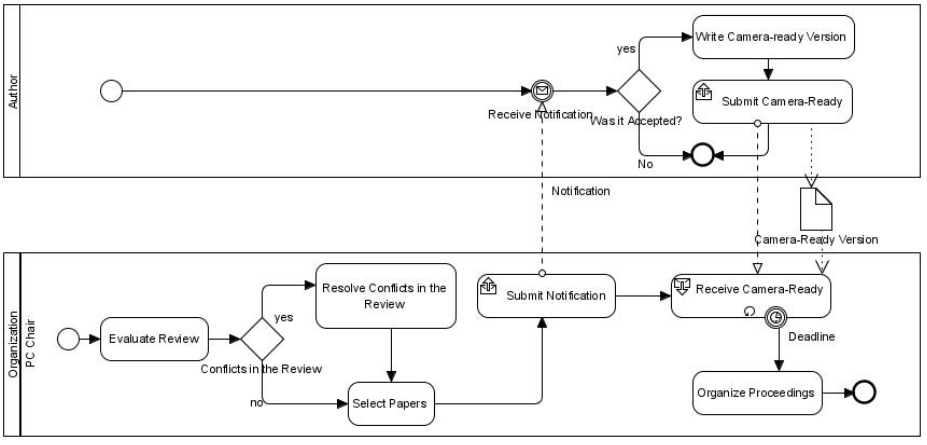

Fig. 2. CMS example

In this process, an Author has previously submitted a paper for a conference and is waiting for the results of the evaluation of his paper. In the conference committee, the PC Chair, which organizes the conference program, is responsible for evaluating the papers reviews and deciding if the paper will be accepted or rejected. Due to space problems we describe only the variants of the task Submit Notification. Due to space problems we describe only the variants of the task *Submit Notification*.

We start with the Step 1.1, performing an analysis of information facets [13] in the task Submit Notification. The task Submit Notification can be performed by the PC Chair or, automatically, by a Conference Management System (CMS). The reception of the notification can include all the authors or be directed just to the first Author. The notification can be sent by email or using a CMS. Finally, the notification submission can be done when a deadline arrives or when all reviews are collected. The identified variations are shown in Table 1.

Analyzing the variations, we will identify what parts of the process will need to be modified to implement each variation - the variation points. We represent the

Table 1. Variants identified for *Submit Notification*

Task	Facet	Variants
Submit Notification	Agentive	PC Chair
		CMS System
	Dative	First Author
		All Authors
	Process	By E-mail
		By publishing in CMS
	Conditional	When the deadline finish
		When all revisions are available

Table 2. Variation Points

Variation Points				
ID	Type	Begin	End	Variants
1	Task	Submit Notification	Submit Notification	1,2
2	Link	Select Papers	Submit Notification	3,4
3	Lane	Author	Author	5,6
4	Lane	PC Chair	PC Chair	5,8

variation points as in Table 2. For each variation point we need to define where they begin and end. These are the points where the variations can be placed. The variation points listed in the Table 2 are the task *Submit Notification* itself, the lanes of *Author* and *PC Chair*, and the link between *Select Papers* and *Submit Notification*.

The variants represented in Table 3 are the same from Table 1, but now with the specification of the type of variants, the dependencies, and the patterns of insertion. In this example, the variant 2 requires the selection of the CMS System as the agent of the task *Submit Notification*, i.e., the variant 2 has a dependency to the variant 5. In the variants 1, 2 and 7, there is a need to change the name of a task, or to replace a task with another one, so their pattern is substitution. The task *Submit Notification* can be substituted by *Submit Notification by E-mail* or *Submit Notification by posting in CMS*.

Now that we know the possible variations in the business process, we are going to define the linking among variants and non-functional requirements. For the sake of space, we are going to consider just two non-functional requirements: Cost and Availability. The aim is to minimize the cost of applying a solution and maximize the availability - i.e., the capacity of readily provide information to the participants of the process. The values of the contribution links varies in a scale from -1 to 1, that means from a negative contribution (increase the cost or damage the availability) to a positive one (minimize cost or maximize

Table 3. Representation of variants

Variants					
ID	Name	Type	Pattern	Dependencies	Variation Point
1	Submit Notify by e-mail	Task	Substitution		1
2	Submit Notify by posting in CMS	Task	Substitution	5	1
3	Deadline	Time Event	Insertion		2
4	All revisions are available	Time Event	Insertion		2
5	CMS System	Lane	Insertion		4
6	Collaborators	Lane	Insertion		3
7	First Author	Lane	Substitution	6	3
8	PC Chair	Lane	Maintain		4

Table 4. Variants and the relationships with Non-functional requirements

Variation Point (ID)	Variants	NFR	
		Cost	Availability
4	PC Chair	0	0
	CMS System	-1	1
3	First Author	0.5	-1
	All Authors	0	0.5
1	By E-mail	0	0
	By publishing in CMS	-1	1
2	When the deadline finish	0	0
	When all revisions are available	0	0

availability). Table 4 presents the result of the linkage between the variants and the non-functional requirements. The values assigned for each variant reveals the impact on Cost and on Availability of the process. For instance, the selection of a CMS as the agent that will perform the notification submission requires the development of a system, which increases the cost of this process. On the other hand, it presents benefits on the process availability, by provinding an accessible environment to share information.

We used the top-down analysis to obtain the instances of the Conference Management process presented in the figures 3 and 4. Figure 3 shows a process configuration that prioritizes Cost, over Availability. It presents the *PC Chair* as responsible for executing the notification submission, and the selected means for doing so is by e-mail. Other variants such as the *Deadline* were included even being neutral if compared with the other variants of the same variation point. The configuration prioritizing Availability (Figure 4) uses some variants that could not be included in the configuration that prioritizes Cost.

4.1 Discussion

The usage of information facets allows a quick elicitation of variability, in contrast to approaches like questionnaires and domain analysis. This is due the pre-defined, objective and limited set of questions that need to be answered during the analysis of information facets.

Our approach is part of an ongoing work. In this way, it may present some limitations. The application of our work may show to be too time consuming, since for every element in the business process we may identify several variations. This effort is multiplied by the number of non-functional requirements being considered. However, this seems to be an inherent problem of any approach that deals with variability, since the amount of variations that may arise in real situations is potentially large. Moreover, we believe that improvements on our approach, for instance, the automation of some of its steps - can minimize this problem. Another limitation is that our approach requires the analyst to have a high expertise on the domain of the process being modeled and to be familiar with the BPMN notation.

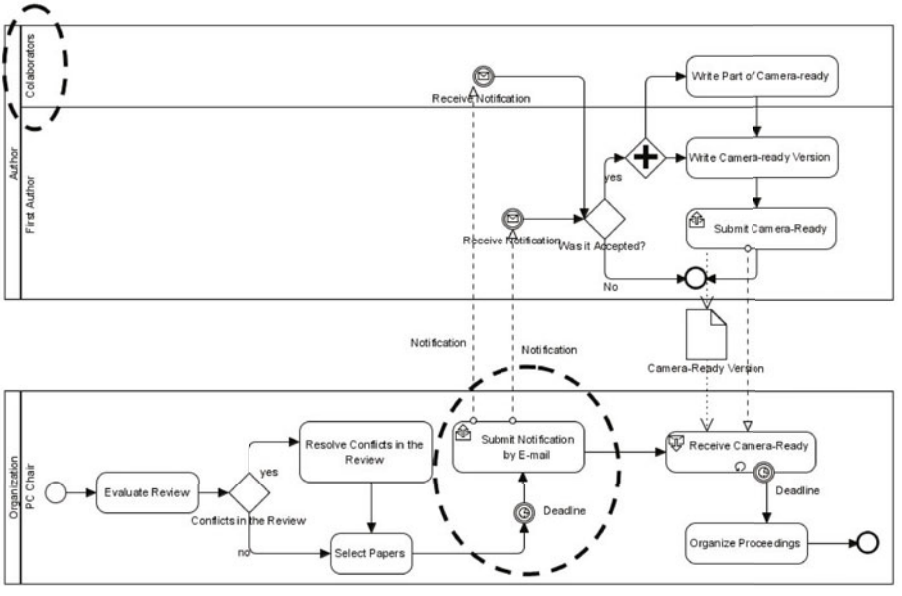

Fig. 3. CMS example with configuration prioritizing Cost

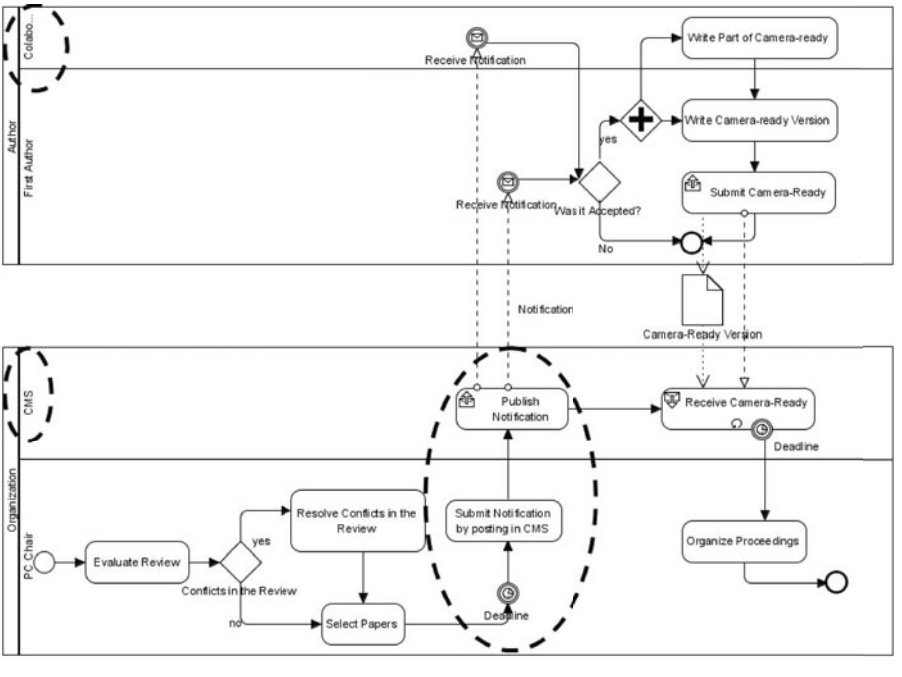

Fig. 4. CMS example with configuration prioritizing Availability

5 Related Work

Schnieders and Puhlmann [2] present a mechanism to represent high variability business process models using BPMN. In this approach they present mechanisms to represent variability in flow-based languages. They rely on extension, inclusion, parameterization and design patterns. These mechanisms enrich the BPMN model and allow the representation of variability with a specific representation for each type of variability. They propose using feature models to obtain the variability but do not explain how to do it. Moreover, their approach is focused on the process itself, without consider the requirements phase. Our approach is not concerned with the representation of variability in the BPMN model itself as Schnieders and Puhlmann do [2]. We believe that the variability represented in an independent model helps the readibility of model.

In Lapouchinian et al. [5] there is an approach that represents business process in terms of its goals. Variability rich business processes are modeled using goal graphs. As the goal graphs are not expressive enough to represent flow and sequence, they apply annotation in the model in order to cover this gap. The aim of this approach is obtain configuration mechanisms that reflect the business process. The result is a configuration mechanism that abstracts the complexity of configuring software from the end-users. Their approach can generate business process (described in BPEL) based transformation of the goal model. We intent to use the non-functional requirements to drive the configuration of models such as Lapouchinian et al. [5]. However by using a generic structure to represent the variability (i.e., matrices) we avoid the work to deal with two types of models (goal and business process models).

Montero et al. [3] describe a methodology to obtain and represent variability in business process models, represented by BPMN language. They are concerned with the derivation of requirements for software related the business process. To represent the business process they adopt feature models and use cases model to describe requirements. The selection mechanism is the selection of features, then if a feature needs to be present in the solution it is selected and the model is restructured to support the changes. As formalism to do it they adopt finite state machines. They select the elements that will be part of the instance by selection of features, we proposed a similar strategy (using bottom-up configuration) but we also allow the configuration using the top-down strategy. Moreover, Montero et al. do not explain why an instance of the business process was selected as we do.

6 Conclusion and Future Work

In this paper we have presented an approach to guide the configuration of business process models using non-functional requirements. This approach spans from the elicitation of variability to the configuration itself, in which instances of the original model are produced. Besides guiding the configuration with clear criteria, this approach also provides the rationale for the selected configuration. In the running example we derived two instances of a conference management

process, each one prioritizing a different NFR: the first one was the instance that resulted in the lower cost, and the second one provided the higher availability.

We consider that the hardest part of this approach is defining the degree of impact of each variant on the NFR. This could be softened through the creation of a catalog that suggests, for each kind of activity in a business process, the impact that activity has on a list of non-functional requirements. This approach can be considered an early activity of the requirements engineering phase, since the resultant process configuration can be used to identify the requirements of an information system to support that process [16]. Since the top down analysis of the configuration can be performed automatically, the system itself could perform a reconfiguration considering context changes that arise during its execution. This behavior is classified as the second level of requirements engineering in dynamic adaptive systems [17].

As future work, we expect to implement supporting tools for this approach. We also intend to improve the priorization used in the top-down analysis, allowing more than one non-functional requirement to be used as criteria, with different weights. Lastly, we are planning to validate this approach performing experimentation with more complex processes. The related works presented in section 5 have parts that are similar or equivalent to parts of our approach. Even if we can not compare the whole approach, we still could compare the similar parts.

Acknowledgments

This work has been partially funded by CNPQ Procs. 143185/2008-0, 565349/2008-2, 308587/2007-3, FACEPE (Grant IBPG-0173-1.03/08), and Project CAPES/DGU Proc. 167/08.

References

1. Halstead, M.H.: Elements of software science. Operating, and Programming Systems Series 7 (1977)
2. Schnieders, A., Puhlmann, F.: Variability Mechanisms in E-Business Process Families. In: Proceedings of 9th International Conference on Business Information Systems, BIS (2006)
3. Montero, I., Peña, J., Ruiz-Cortés, A.: Representing Runtime Variability in Business-Driven Development Systems. In: Proceedings of the Seventh International Conference on Composition-Based Software Systems (ICCBSS 2008), pp. 605–608. IEEE Computer Society Press, Los Alamitos (2008)
4. La Rosa, M., van Der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: Questionnaire-based variability modeling for system configuration. Software and Systems Modeling 8(2), 251–274 (2009)
5. Lapouchnian, A., Yu, Y., Mylopoulos, J.: Requirements-Driven Design and Configuration Management of Business Processes. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 246–261. Springer, Heidelberg (2007)

6. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering, vol. 5. Kluwer Academic Publishers, Dordrecht (2000)
7. Cardoso, E.C.S., Almeida, J.a.P., Guizzardi, G., Guizzardi, R.S.S.: Eliciting Goals for Business Process Models with Non-Functional Requirements Catalogues. In: Enterprise, Business-Process and Information Systems Modeling, pp. 33–45. Springer, Heidelberg (2009)
8. Pavlovski, C.J., Zou, J.: Non-functional requirements in business process modeling. In: APCCM 2008: Proceedings of the fifth on Asia-Pacific conference on conceptual modelling, pp. 103–112. Australian Computer Society, Inc. (2008)
9. Aburub, F., Odeh, M., Beeson, I.: Modelling non-functional requirements of business processes. Information and Software Technology 49(11-12), 1162–1171 (2007)
10. OMG, C.O.: Business Process Model and Notation (BPMN) - Specification v1.2 (2009)
11. White, S.A., Miers, D.: BPMN Modeling and Reference Guide: Understanding and Using BPMN. Future Strategies Book Division (2008)
12. Pohl, K., Bockle, G., Linden, F.V.D.: Software product line engineering, vol. 49. Springer, Heidelberg (2005)
13. Liaskos, S., Lapouchnian, A., Yu, Y., Yu, E., Mylopoulos, J.: On Goal-based Variability Acquisition and Analysis. In: Proceedings of 14th IEEE International Conference Requirements Engineering, RE 2006, pp. 92–96 (2006)
14. Wohed, P., van Der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M., Russell, N.: Pattern-based Analysis of BPMN (2005)
15. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Reasoning with goal models. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, pp. 167–181. Springer, Heidelberg (2002)
16. De La Vara, J.L., Sanchez, J., Pastor, O.: Business Process Modelling and Purpose Analysis for Requirements Analysis of Information Systems. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 213–227. Springer, Heidelberg (2008)
17. Berry, D.M., Cheng, B.H.C., Zhang, J.: The four levels of requirements engineering for and in dynamic adaptive systems. In: 11th International Workshop on Requirements Engineering Foundation for Software Quality (REFSQ), p. 5 (2005)