Process Variability: concepts, approaches and its application on a model of Cloud BPM

Jose Martinez Garro[1], Patricia Bazan[1], Javier Diaz[1]

[1] LINTI, Facultad de Informática, UNLP,
La Plata, Argentina
{josemartinezgarro, pbaz, jdiaz}@info.unlp.edu.ar

Abstract. Business Process Management as a discipline has suffered several changes during the implementation of the execution and monitoring phases in the cloud model. Different strategies have been seen in terms of the implementation needed in order to gather information from the different nodes during process execution, and finally show the results seamlessly without the notion of a partitioned business process. Another aspect to introduce in this context is Process Variability, in terms of the changes suffered by a process model during its lifecycle, and how these changes affect the actual instances in execution. In terms of a cloud BPM implementation, Process Variability adds even more complexity during execution considering the different process portions, as well as during the gathering and monitoring phases. The main purpose of this work is to establish how the different aspects of a cloud BPM implementation with decomposed processes are affected by introducing concepts of Process Variability, both in execution as well as in the monitoring phase. To achieve this goal an analysis of some current bibliography and the main aspects of process variability management is accomplished.

Keywords: BPM, Cloud, Execution, Monitoring, Process Variability.

1 Introduction

BPM (*Business Process Management*) has presented a pronounced growth during the last years, provoking an exploration over different related technologies, such as the cloud model in terms of execution, or CEP (*Complex Event Processing*) for process monitoring. Another aspect that must be considered during process execution and monitoring is Process Variability. Most of the non-open source products have features to support process variability, but in terms of a single BPM node. Since the cloud orientation is nowadays a trend in mostly every software paradigm, including BPM, it is necessary to consider how the management of process variations could affect the process decomposition and execution in the cloud. Process variability refers essentially to the different variations that a process model could suffer during its lifecycle, and how these changes could affect the instances in execution, and in consequence the monitoring of them. In chapter 2 an analysis of the current status of process variability and formal verification is accomplished. After that, in chapter 3 a brief revision of the mechanisms needed for process variant handling are evaluated. In chapter 4 the requirements for a formal specification are presented, and the different

policies and actions in declarative and imperative variability are detailed. Finally in chapter 5 it is addressed the way on how all these previous concepts affect a concrete implementation of Cloud BPM presented in some previous works by the same authors of this work [22] [23][29].

2 State of the art: variability and formal verification

There are several related works ([1], [2], [3]) addressing these two concepts, especially when it comes to process models and version management. In terms of definition, a formal verification mechanism implies proving or disproving the correctness of a system model according to a formal specification using some formal methods of mathematics. When employing formal verification, a system model – often represented by a labeled transition system – is verified against a formal specification using logic enunciations. One approach towards formal verification is model checking. When this task is performed, a system model is automatically, systematically, and exhaustively explored while each state is verified for compliance with the formal specification. In this way, Business Process Verification is the act of determining whether a business process model complies with a set of formal correctness properties [1] [2] [3] [29].

2.1 Soundness

Business process correctness verification entails the verification of a set of basic properties such as reachability and termination. Reachability applied to a business activity requires an execution path from the starting activity to every other activity in the model. The termination property requires that all possible execution traces reach a final state. Business process soundness, a property originally proposed in the area of Petri Net verification, is known as the combination of these two properties adding a third one: the absence of related running activities at process termination (i.e., proper completion) [2] [4] [5] [6].

2.2 Compliance

Business process compliance aims to confirm that a business process adheres to a set of rules imposed on that process. Rules can, for example, be imposed upon a process by international regulations, national law, or internal business rules. Whereas soundness verification aims at the verification of a limited set of requirements like reachability, termination, and possibly proper completion – compliance verification requires the verification of a broad set of specifications [1] [3] [7] [8] [24].

2.3 Variability

BPM is evolving rapidly due to emerging mass customization and personalization trends, the need for adaptation to varying business and execution contexts, and a wider availability of service-based infrastructures. Variability is an abstraction and management method that addresses a number of related issues, especially in a cloud environment where different models are deployed constantly over different nodes of the architecture. Variability can be introduced to the BPM area by using imperative or declarative approaches. Whereas imperative approaches exactly specify possible changes, declarative approaches constrain the process behavior, allowing any change within those constraints. In addition, since imperative approaches exactly specify all possible changes, they require all of them to be known in advance. Conversely, declarative approaches do not require such knowledge [9] [10] [21] [22].

3 Dealing with process variants

There are different ways to think solutions for managing variants in existing BPM tools according to the current bibliography, and they can be divided into two approaches: the multi-model and the single-model approach. In this section some concepts evaluated in a few related works are presented according to how they address all this terms in traditional BPM.

3.1 Multi-model approach

In existing BPM tools, process variants often have to be defined and kept in separate process models. Typically, this results in highly redundant model data as the variant models are identical or similar for most parts. Furthermore, the variants cannot be strongly related to each other; i.e., their models are only loosely coupled (e.g., based on naming conventions). As a conclusion, generally, modeling all process variants in separate models does not constitute an adequate solution for variant management [2] [3] [11].

3.2 Single model approach

Another approach, frequently applied in practice, is to capture multiple variants in one single model using conditional branchings (i.e., XOR-/OR-Splits). Each execution path in the model represents a particular variant. Therefore, branching conditions indicate which path belongs to which variant. Generally, specifying all variants in one process model could result in a large model, which is difficult to comprehend and expensive to maintain. Neither the use of separate models for capturing process variants nor their definitions in a model based on conditional branching constitute an adequate method. Both approaches do not treat variants as first class objects; i.e., the variant-specific parts of a process are maintained and hidden either in separate models

(multi-model approach) or in control flow logic modules (single-model approach) [1] [3] [12] [13].

3.3 Lifecycle

In terms of Process Variants, the standard process lifecycle consists of three phases (Fig. 1), namely the design and modeling of the process, the creation of a particular process variant, and the deployment of this variant in a runtime environment (according to the selected approach).

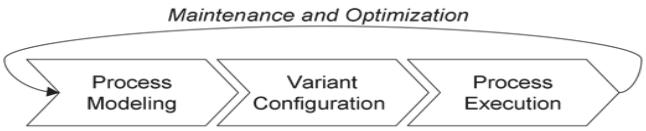

Fig 1: Process Variant lifecycle [2]

Modeling: the efforts for modeling process variants should be kept as minimal as possible, so in this direction, reusing variant models (or parts of them) has to be supported. In particular, it should be possible to create new variants by taking over properties from existing ones, but without creating redundant or inconsistent model data.

Variant configuration: the configuration of a process variant (what essentially means its derivation from a given master or base process) should be done automatically if possible.

Execution: to execute a process variant, its model has to be interpreted by a workflow engine, so in this context, it is important to keep information about the configured process variant and its relation to a master or base process (and to other variants) in the runtime system.

Maintenance and optimization: in order to reduce maintenance efforts and cost of change, fundamental updates affecting multiple process variants should be conducted only once [2] [3] [14] [15].

4. Applying verification in Process Variability. The challenge in Requirements Specification

In terms of process verification, as we have seen previously in Section 2, there are three main aspects to be considered as well: soundness, compliance and variability. This last one builds upon the concept of compliance. In the context of BPM, variability indicates that parts of a business process remain variable, or not fully defined, in order to support different versions of the same process depending on the

intended use or execution context. Imperative variability employs the use of variation points to provide different options at certain locations inside the process. Declarative variability uses specifications like those of compliance to specify how each version of a process should behave and absorb the new requirements [1] [21] [22] [29].

4.1 Imperative Variability

Imperative structural adaptation consists of atomic operations which, when executed in a specific predefined sequence, rearrange a business process to form a specific variant. Table 1 shows some mechanisms commonly known as Atomic Structural Adaptations.

Table 1: Imperative variability

Object	Action
Process Fragment	Insert/Delete/Move/Replace/Swap/Copy/Embed in loop/Parallelize/Embed in traditional branching
Subprocess	Extract/Inline
Control dependency	Add/Remove
Other	Update condition

Continuing with this idea, it also should be able to express the following self-explanatory atomic resource adaptations to the business process. These features are commonly named as Atomic Resource Adaptation. A variability management framework should at the same time allow the process designer to express the above imperative structural adaptation requirements, which are commonly named as Variation Relations [1] [3] [16] [17].

4.2 Declarative Variability

Declarative specifications consist of a set of rules expressing variations by acknowledging the borders which limit the possible process modifications. They are useful in order to set boundaries at the time of modifying a process model. Unlike atomic structural changes which indicate imperatively what can vary, a declarative specification limits the borders of changes explicitly [2] [18] [19] [20].

5 Applying Process Variability policies to Cloud BPM

As it was presented previously, there are different approaches and mechanisms to handle process variants in BPM. In our case, it is important to perceive that managing variability in a cloud environment amplifies every issue presented for single-tenant scenarios. In case of applying this approach to process monitoring, during the gathering phase, it is important for each instance to feed the monitoring server (i.e. using complex events in CEP or BAM), no matter the version of the process they belong to [1] [2] [22] [24].

In practice, process variants are often created by cloning and adjusting an existing process model of a particular type according to the given context. Generally, every process model can be derived out from another one by adjusting it accordingly, i.e., by applying a set of change operations and change patterns, respectively, to it. There are different implementations of process variability, and each framework applies policies according to its own particularities. One of these frameworks present in current bibliography is Provop ([2], implemented for single tenant BPM servers). Starting from this observation, Provop provides an operational approach for managing process variants based on a single process model. In particular, process variants can be configured by applying a set of high-level change operations to a given process model. The latter one is denoted as the base process [25].

5.1 Modeling

In the modeling phase, first of all, a base process, from which the different process variants can be derived through configuration, has to be defined. Following this, some high-level change operations, which can be applied to this base process, are specified. There is a thing that results fundamental for configuring a process variant: the base process. This serves as reference for the high-level change operations. Basically, the approach (e.g. Provop) should support policies that consider the standard process, the most frequently used process, the minimal average distance between a model and its variants, the superset of all process variants and the intersection of all process variants. This framework should also consider change operations, grouping change operations into options**,** constraint-based use of options and the context model [1] [3].

5.1.1 Policies application in a Cloud environment

As it was previously presented in [29], in a cloud based decomposed solution there are several process partitions that together conform the original process model. In this scenario, it is necessary to coordinate different models that in combination compose the original base process. In case of adding to this some process variability features, it could be necessary to apply a particular policy during defining the base process (or even more than one) in a particular portion of it, and the other ones could be not affected by the change [2] [21] [22] [23] [29].

5.2 Variant Configuration

In the configuration phase, the base process, the options defined for it and the context model are used to configure the models of the different variants. More precisely, a particular variant is configured by applying a sequence of options and their corresponding change operations to the base process. The sequence of steps is given as (1) select the relevant options, (2) evaluate relations between selected options, (3) determine the order in which options shall be applied, (4) apply options and their change operations and checking consistency [1] [2] [26].

5.2.1 Configuration in a cloud based environment

Once again following the same line as in [29], in a cloud based model the complexity of each step application gets amplified because of the existence of several decomposed models forming the original base model. In summary, option constraints are considered to ensure semantic correctness and consistency of the selected set of options at configuration time. This follows from the above described policies for defining the base process, assuming, for example, a base decomposed process being defined as an intersection of its variants [2] [22] [23] [29].

5.3 Deployment and Execution

After the configuration phase, the resulting variant model needs to be translated into an executable workflow model. Common tasks emerging in this context are to assign graphical user interfaces, to subdivide workflow activities into human and automated tasks, or to choose the right level of granularity for the workflow model [2] [26].

5.3.1 Deployment in a cloud based environment

According to the scenario previously presented in [22] and [29], and using the same architecture that integrates several nodes with Bonita in a private cloud, to apply all these concepts in the execution and deployment of decomposed process could result a very intricate task. As it was seen in previous works, each part of the process is in charge of invoking the next one with the goal of maintaining the original process flow, and for this invocation some vital information is needed: server direction, version of the process model, user and password to connect with the API [2] [29].

5.4 Maintenance and Optimization

When evolving base processes (e.g., due to organizational optimization efforts or changes in the business rules), all related process variants (i.e., their models) must be reconfigured automatically. Thus, maintenance efforts can be significantly reduced [27] [28].

5.4.1 Maintenance in a cloud based environment

Evolving and optimizing the base process may affect existing options, for example, when the referred adjustment points are moved to a new position or are even deleted. These actions cause that processes loose reference points and then certain actions are more complicated, for example checking whether the definitions of existing options are affected by the adaptations of the base process model.

As we have seen previously in [29], there are methods like BAM (*Business Activity Monitoring*) or even CEP (*Complex Event Processing*) used in a distributed environment in order to obtain relevant information about the process in execution. Table 2 shows how the different policies and actions could be added to a cloud decomposed model in order to handle process variant management during the whole lifecycle [1] [2] [29].

Lifecycle Phase	Policy/Action
Modeling	**Definition**: standard process - most frequently used process - minimal average distance - superset of all process variants - intersection of all process variants. change operations, grouping change operations, constraint-based options and context model
Variant configuration	Select relevant options - Evaluate relations between selected options - Determine the order in which options shall be applied - Applying options and their change operations - Checking consistency
Deployment and Execution	Assign graphical user interfaces - Subdivide workflow activities into human and automated tasks - Choose the right level of granularity for the workflow model
Maintenance and optimization	Automated reconfiguration

Table 2: Policies and actions to apply in cloud BPM

6. Conclusions

BPM since the beginning was conceived as a methodology capable of reducing the gap between the market and the final implementation of the business processes that interact with it. The conditions affecting a process may change any time, so the rules within the organization, causing in fact that every process definition should be adapted according to some new specifications.

When a BPMS is already inserted in the organization and each process has instances in execution, to generate a process variant is not a simple task: the process analyst should decide how to apply the changes in the base model, how to promulgate them in the existing instances, if every instance is going to be affected by the changes or not, and finally being capable of manage the different versions of the process in parallel. If this does not seem simple even in a single tenant environment, neither it is in a cloud

decomposed one where each process runs separately, in different servers united by an execution chain provided by an API in the BPMS. There are several frameworks in the current bibliography for process variant handling (like Provop as it was named repeatedly in previous sections), tending to implement different policies in order to apply changes in process instances and automate different tasks that enhance process variant management and maintenance. A possible line for future works is to implement a concrete version of a framework (such as Provop) that implements the automation of process variants promulgation in a cloud environment considering decomposed processes that need to be chained during execution and monitoring.

References

1. M Reichert, A Hallerbach, T Bauer. “Lifecycle Management for Business Process Variants”. University of Ulm, Daimler TSS GmbH, Neu-Ulm University of Applied Science, Neu-Ulm, Germany. March 2015

2. G, Heerko. “Business Process Variability: A Study into Process Management and Verification”. Rijksuniversiteit Groningen. 2016

3. B Estrada-Torres, A del Río-Ortega, M Resinas and A Ruiz-Cortes. “Identifying Variability in Process Performance Indicators”. Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Seville, Spain. March 2017

4. SM Reza Beheshti, B Sherif Sakr, D Grigori, H Nezhad, M Ahmed Gater, S Hwan Ryu. “Process Analytics: Concepts and Techniques for Querying and Analyzing Process Data”. Springer International Publishing. Switzerland 2016.

5. G Castro Barbosa Costa, C M.L. Werner and R Braga. “Software Process Performance Improvement Using Data Provenance and Ontology”. Systems Engineering and Computer Science Department, Federal University of Rio de Janeiro COPPE, Rio de Janeiro, RJ, Brazil. August 2016.

6. V Ferme, A Ivanchikj and C Pautasso. “Estimating the Cost for Executing Business Processes in the Cloud”. Faculty of Informatics, USI Lugano, Lugano, Switzerland. August 2016

7. M Hewelt and M Weske. “A Hybrid Approach for Flexible Case Modeling and Execution”. Hasso Plattner Institute Potsdam, Potsdam, Germany. August 2016

8. D Sanchez-Charles, V Muntes-Mulero, J Carmona, and M Sole. “Process Model Comparison Based on Cophenetic Distance”. CA Strategic Research Labs, CA Technologies, Barcelona, Spain. August 2016

9. T Lehto, M Hinkka and J Hollmen. “Focusing Business Improvements Using Process Mining Based Influence Analysis”. QPR Software Plc, Helsinki, Finland. August 2016

10. A Mos and M Cortes-Cornax. “Business Matter Experts do Matter: A Model-Driven Approach for Domain Specific Process Design and Monitoring”. Xerox Research Center, 6 Chemin de Maupertuis, Meylan, France. August 2016

11. B Karim, Q Tan, I El Emary, B A. Alyoubi, R Soler Costa. “A proposed novel enterprise cloud development application model”. Springer-Verlag Berlin Heidelberg 2016

12. O Skarlat, M Borkowski and S Schulte. “Towards a Methodology and Instrumentation Toolset for Cloud Manufacturing”. Distributed Systems Group, TU Wien. April 2016

13. E Hachicha, N Assy, W Gaaloul and J Mendling. “A Configurable Resource Allocation for Multi-Tenant Process Development in the Cloud”. Telecom SudParis, UMR 5157 Samovar, Universite Paris-Saclay, France. December 2015

14. G Rosinosky, S Youcef, F Charoy. “An Eficient Approach for Multi-tenant Elastic Business Processes Management in Cloud Computing environment”. 2016 hal-01300188

15. A Gunka, H Kuehn and S Seycek. “BPM in the Cloud: The BOC Case”. BOC Information Technologies Consulting GmbH, Vienna, Austria. 2017

16. G Rosinosky, S Youcef, F Charoy. “A Framework for BPMS Performance and Cost Evaluation on the Cloud”. Workshop "Business Process Monitoring and Performance Analysis in the Cloud", Dec 2016, Luxembourg, Luxembourg.

17. N. Herzberg, A. Meyer, M. Weske. “An Event Processing Platform for Business Process Management”. Business Process Technology Group, Hasso Plattner Institute at the University of Potsdam. Potsdam, Germany. June 2013.

18. S. Bulow, M. Backmann, N. Herzberg, T. Hille, A. Meyer, B. Ulm, T. Y. Wong, M. Weske. “Monitoring of Business Processes with Complex Event Processing”. Business Process Technology Group, Hasso Plattner Institute at the University of Potsdam. Potsdam, Germany. July 2013

19. C. Zeginis, K. Kritikos, P. Garefalakis, K. Konsolaki, K. Magoutis and D Plexousakis. “Towards Cross-Layer Monitoring of Multi-Cloud Service-Based Applications”. Institute of Computer Science Foundation for Research & Technology – Hellas. Grece. August 2013.

20. M. Goetz. “Integration of Business Process Management and Complex Event Processing”. iTransparent GmbH, IT Consulting, Bergstraße 5, 90403 Nuremberg, Germany. November 2010.

21. J Martinez Garro, P Bazán. “Monitoreo de procesos en el cloud. Una propuesta arquitectónica”. JCC 2013. Universidad de Temuco. Chile. November 2013.

22. J Martinez Garro, P Bazán. “Constructing and monitoring processes in BPM using hybrid architectures”. IJACSA Journal. Londres. Febrero 2014.

23. J Martinez Garro, P Bazán, J Diaz. “Decomposed processes in Cloud BPM: techniques for monitoring and the use of OLC”. WORLD COMP 2014. Las Vegas, USA, July 2014.

24. J Martinez Garro, P Bazán, J Diaz. “OLC y Monitoreo de procesos en el cloud: un caso de estudio”. JCC 2014. Chile. November 2014.

25. R Confortia, M La Rosaa, G Fortinoc, A H. M. ter Hofstedea, J Reckera, M Adamsa. “Real-Time Risk Monitoring in Business Processes: A Sensor-based Approach”. Queensland University of Technology, Brisbane, Australia. May 2013.

26. E Mulo, U Zdun, S Dustdar. “Domain-Specific Language for Eventbased Compliance Monitoring in Process-driven SOAs”. Distributed Systems Group Institute of Information Systems Vienna University of Technology, Vienna, Austria. April 2013.

27. P Szwed, W Chmiel, S Jedruzik, P Kadluczka, “Business Process in a Distributed Surveillance System integrated through workflow”. Automatika. Vol 17. No 1. November 2013.

28. V Stavrou, M Kandias, G Karoulas, D Gritzalis. “Business Process Modeling for Insider Threat Monitoring and Handling”. Information Security & Critical Infrastructure Protection Laboratory Dept. of Informatics, Athens University of Economics & Business. Greece. May 2014.

29. J Martinez Garro, P Bazan, J Diaz. “Using BAM and CEP for Process Monitoring in Cloud BPM”. JCST Cloud Journal. April 2016