A declarative approach for change impact analysis of business processes

Adeel Ahmad, Henri Basson, Mourad Bouneffa, M Matsuda

A declarative approach for change impact analysis of business processes

Adeel Ahmad[1], Henri Basson[1], Mourad Bouneffa[1], and M. Matsuda[2]

[1] LISIC, Université du Littoral Côte d'Opale, EILCO, France
[2] Kanagawa Institute of Technology, Japan
[1]{adeel.ahmad, henri.basson, mourad.bouneffa}@univ-littoral.fr
[2]matsuda@ic.kanagawa-it.ac.jp

Abstract. The business process models provide a means to control and visualize the enterprise processes. Different processes in an enterprise inter-operate to achieve a common strategic and operational objective. These processes continuously evolve to meet the changing business requirements. In this respect, the process models should be able to reflect a cost-effective solution for the decided changes in a process and its impact on other executing processes. Such dynamic adaptability requires not only an exhaustive comprehension of business process activities but also the understanding of the various change dimensions. In this work, we propose a formal description of change feasibility, change incorporation, and traceability of the change impact propagation among multiple processes. A rule-based approach is proposed for change incorporation during the development and instantiation of business process models. The rule-based declarative approach is destined to estimate the change feasibility in dynamic business process models. We attempt to analyze the multiple dependency levels to better control the change impact propagation. The work aims to help a well-controlled and successful evolution of business processes.

Keywords: Business process modeling, Business process evolution, Change impact analysis, Rule based change management, Structural dependencies, Data dependencies.

1 Introduction

Business Process Models (BPM) follow a continuous cycle of process discovery, process modeling, deployment, execution, improvement, and redesign [1, 2]. However, it is generally observed that the enterprises are reluctant to change the existing BPMs [3-5] because of the associated complexity and the cost. Indeed, the evolution of inter-operable business processes can generate difficult situations for the creation, modification, or deletion of process fragments in the rectified schemas. This problem can further aggravate when the instances of concerned process fragments are already in execution while introducing the change. It is because of the compliant of business process instances with the definition of their types, i.e., whether a respective change can correctly propagate its impact without causing inconsistencies or errors (e.g. deadlocks,

live-locks) [6]. This can result a non-compliance with regulations [7] or a degradation of the quality of the business process [8, 9].

The changes at process instance level (also known as instance-specific changes) are often applied in an *ad-hoc* manner to deal with the exceptions (unanticipated situations) resulting in an adapted instance-specific process schema [10]. These are specific to a particular instance, which means changes in one instance usually do not affect other running process instances. In many cases, changing the state of a process instance is not sufficient for a successful BPM evolution; the process structure itself has to be adapted as well [11]. For this reason, the change at the process type level (also named as process schema evolution) is necessary to deal with the evolving nature of process roles (e.g., to adapt them to new legal requirements or new policies). The schema evolution often leads to the propagation of respective changes to the rest of the schema components and also to the ongoing process instances. This is particularly true if the instances have a longer runtime (e.g., medical or handling of leasing contracts, *etc.*).

The rest of the paper is structured as follows: the section 2 provides a brief overview of the related work. We explain, in detail, the dependency relationships and their analysis in section 3. Whereas, the section 4 describes the assessment of the change feasibility and the analysis of the impact propagation of dynamic changes with the help of rules. We briefly discuss implementation prototype in section 5. Later in section 6, we conclude the content of this article.

2 Related work

The research on change management of business processes has been continuing to attract increasing interest from the industry and the scientific community in the last couple of decades. The major focus remained on integrating changes into business processes without affecting running instances. While, it is observed that an *a priori* analysis of the change impact is given less consideration.

Several approaches and paradigms [12-16] have been proposed to cope with the changing processes and their flexibility. In [12], the authors suggest an algorithm to calculate the minimal region affected by the changes that is based on Petri-Nets. It attempts to identify the change regions to check the compatibility of workflow changes. In [13], authors discuss, a formal approach based on the notion of process constraints called Constraint-Based Flexible business process management. It has been developed to demonstrate, how the specification of selection and scheduling constraints can lead to increased flexibility in process execution, while maintaining a desired level of control. Similarly, the authors in [14] propose a combination of a set of change patterns and seven *change support features* dealing with the process change. In this regard, *YAWL* [15] is an initiative based on formal foundations that shows significant promise in the support of a number of distinct flexibility approaches. Also *Declare* [16], in this regard, offers to examine the change; its declarative basis provides a number of flexibility features. Interestingly, it supports transfer of existing process instances to the new process model.

In [17], the author suggests a flexible modeling and execution of workflow activities based on a business meta-model. This approach supports dynamic changes such as adding or deleting activities, but requires that the activity is not in the running state when incorporating the change.

Apart from the work listed above, in [18] the authors attempt to analyze the dependency relationships that exist within a workflow. However, their focus has been constrained on modeling the workflow rather than on the change impact analysis, and most of the dependency relationships are confined to the structural dependencies, i.e. intra-dependency of activity or routing.

In [19], the author presents a framework to analyze four types of dependencies concerning the activities, roles, data, and actors. The objective of this framework is limited to use this analysis to generate a set of "transition conditions" which are deployed in a distributed process control. The work of authors in [19] is closely relevant to our proposition. It uses the dependency analysis for the purpose of change impact analysis and suggests using a set of queries defined in PROLOG[1] to help designers and business experts to understand the dependencies between different elements of the business process model.

The use of rules makes the approach more general compared to the algorithms. We believe, the declarative rules can help to determine the feasibility and assess an *a priori* change impact in multiple business process modeling languages (e.g. BPMN, EPC, UML Activity diagrams, etc.).

3 Analysis of dependency relationships

We attempt to establish a scalable base to progressively consider the different inter-dependent dimensions of process models such as activities, data, actors, resources, etc.

Our objective is to identify the potentially affected elements for an *a priori* change impact analysis in the evolving business processes ahead the change implementation. We should consider the critical dependencies that may exist between the process model artefacts such as activities, data, roles, actors, resources, events, services, and rules, etc. In this paper, we specifically focus more on the multi-dimensional business process dependency model to get an insight concerning different dependency relationships among business processes.

In the following, we formally discuss some of the major dependency relationships in business processes.

3.1 Activity dependency (routing)

The activity dependency reflects the execution order of the business process activities. This ordering is usually defined by the modelers or business experts. It is based on technical requirements, legal regulations, and management policies. For example, if

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

two activities are executed sequentially, it means that the completion of the execution of the first activity is a *pre-condition* for the execution of the second.

The activity dependency shows the execution order of activities within a business process through the control-flows i.e. *sequence flow* and *message flow*. This dependency defines not only the execution order but also the semantics associated with this ordering. For example, for an *AND-Join* routing of three-activities *A*, *B* and *C*; *A* and *B* must be executed before *C* (furthermore, in synchronization either *A* or *B* must finish before the *C* can start its execution, *etc.*).

An activity dependency is formally defined as: $D_a = (D_p, \Omega)$ over a set of activities $A = \{a_1, a_2, a_3, \ldots, a_n\}$ and a set of control-flows $T = \{t_1, t_2, t_3, \ldots, t_n\}$, where:

$$D_p = D_pi(a) \cup D_po(a) \text{ where } a \in A. \quad (1)$$

The $D_pi(a)$ is a set of all preceding activities $a_i \in A$ (denoted as: $a \rightarrow a_i$) on which the execution of activity *a* is dependent. The relationship can be a *many-to-one*, i.e., one activity depends on multiple activities.

In the same way, $D_po(a)$ is a set of all succeeding activities $a_i \in A$ (denoted as: $a_i \rightarrow a$) meaning their executions depend on the activity *a*. The relationship can be *one-to-many* i.e., multiple activities depend on one activity.

$$\Omega = \Omega_i \cup \Omega_o \quad (2)$$

The Ω_i is a set of control-flows, $t_i \in T$, connecting each activity $a_i \in D_pi$ to *a*, i.e. all incoming arcs $(D_pi(a),a)$ of *a*.

The Ω_o is a set of control-flows, $t_i \in T$, connecting *a* to each activity $a_i \in D_po$, i.e. all outgoing arcs $(D_po(a),a)$ of *a*.

3.2 Role dependency

The role is a logical abstraction of one or more actors, usually in terms of common responsibility or position. It means an actor can be a member of one or more roles. It is observed, that a role is always associated to some activities.

The role dependency can be described through a role-net, which can be achieved by replacing the roles to the activities associated with them. In other words, the activity-based flowchart becomes a role-based flowchart while at the same time dependency relationships depend on routing entities.

For further clarification of the role dependency, let us consider a role R_1, which can be assigned to the same activities that are being executed by another role R_2, then the role R_2 may have a dependency relationship with the role R_1.

For the sake of further clarity, let us consider the activity "*blood test*" in a review process for medical checkup. It can be performed by a nurse or doctor. That is, the *role (nurse, blood test, medical checkup)* and *role (doctor, blood test, medical checkup)* are assigned to same activity which is "*blood test*". Therefore, there exists a dependency relationship between the role *nurse* and the role *doctor*.

If we consider $R = \{r_1, r_2, r_3, \ldots, r_n\}$ as a set of roles and $A = \{a_1, a_2, a_2, \ldots, a_n\}$ as a set of activities then the role dependency can be formally represented as:

$$D_r = (\sigma, \Psi) \text{ where, } \sigma(r) = \sigma_i(r) \cup \sigma_o(r), \text{ and } r \in R. \quad (3)$$

The $\sigma_i(r)$ represents the set of roles which are immediate predecessors of role r, i.e. these are the roles which are affected to the activities a_i where the activities $a_i \in A$ precede the activity a (for r associated to a). The $\sigma_o(r)$ represents the set of roles which are the immediate successors of role r. These are the roles which are affected by the activities a_i where the activities $a_i \in A$ succeed the activity a (for r associated to a).

$$\Psi = \Psi_i \cup \Psi_o \quad (4)$$

The Ψ_i is the set of control flows ($t_i \in T$) or the arcs related to each role of $\sigma_i(r)$ to the role r, i.e. the set of incoming arcs of role r. In the same way, the Ψ_o represents the set of control flows ($t_i \in T$) linking the role r to all the roles of the $\sigma_o(r)$, i.e. the set of outgoing arcs of role r.

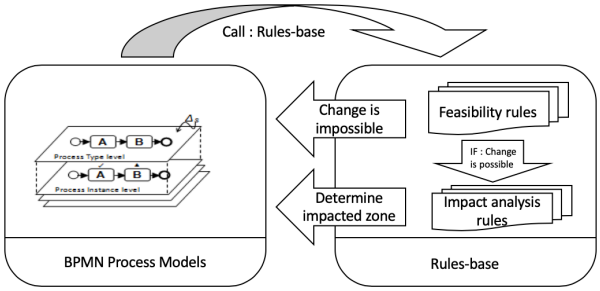

Fig. 1. Change impact assessment with the help of rules.

4 Declarative assessment of change impact

In the presented approach, as broadly described in Fig. 1, the impact propagation is assessed with the help of rules written in *ECA* or <Event> <Condition> → <Action> formalism. It encompasses two steps, which are explained below:

1. Assess the feasibility of the dynamic change in BPMN process model with the help of a set of rules called feasibility rules.
2. If the change is feasible, then perform an *a priori* analysis of the impact propagation at the process type level and in the corresponding instances with the help of a set of rules called impact analysis rules.

The change operations can be a combination of addition, deletion, or modification of activities, but these can also become more complex depending on their abstraction and granularity. The complex change operations can involve the replacement of a process fragment by another one, moving a process fragment from its current position in the flow to a new one, copying a process fragment, swapping a process fragment with another, parallelization of process fragments, or some other complex action. The meta-model of change impact analysis, as shown in Fig. 2 encompass the possible prospects of the change. This provides a useful overview of the different concepts concerning the

change and types of impacts to support the business process change impact analysis. Any change of a business process can propagate a multi-faceted impact i.e. structural, functional, behavioral, logical, and qualitative impacts. Therefore, it leads to a comprehensive analysis as required by its definitions.

In the following, we formally describe the change impact analysis in business processes. A change operation can consequently result in a difference (denoted as Δ), between the initial process schema S_0 and the modified process schema S_1. This can be expressed as follows:

$$S_1 = S_0 + \Delta \tag{5}$$

$$\Delta = | S_1 - S_0 | \tag{6}$$

The variant (Δ) can generate the post-change impacts on whole or part of the process model and its running instances. Therefore, an *a priori* analysis of this variant is important to ensure the correctness and consistency of the change impact propagation. Otherwise, changes such as the deletion or the addition of a task may cause severe inconsistencies (e.g., unintended update loss) or even run-time errors (e.g., program crashes due to the invocation of task modules with invalid or missing parameters).

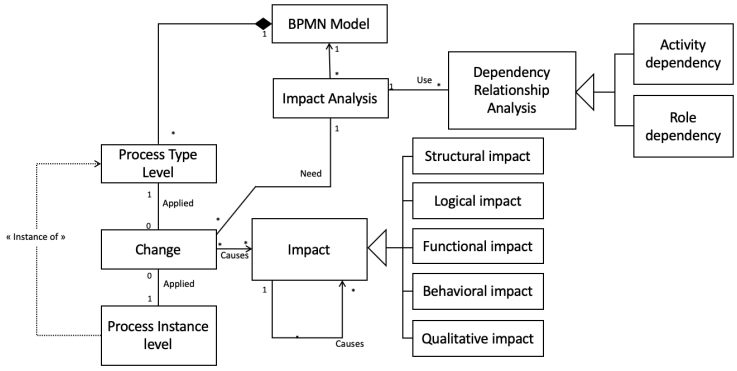

Fig. 2. Meta-model of change impact analysis.

4.1 Feasibility Rules

The set of *Feasibility Rules (FR)* ensures the compliance of business process instances to the definition of their type during a change. It can be used to assess the feasibility of the dynamic changes. To further illustrate, let us consider, as described below, the example of a process type level change.

The rule *process type level change* ensures the feasibility of the dynamic change at process type level. It is defined as follows:

- In order to avoid the insertion of a new task *T* as a predecessor of an already *RUNNING* or *COMPLETED* task, we require that all the succeeding elements in the control flow must be in one of the states as *NOT_ACTIVATED* or *ACTIVATED*. Conversely, the preceding tasks may be in an arbitrary state.

- The deletion of a task *T* of a running process instance is only possible, if *T* is either in *NOT_ACTIVATED* or in *ACTIVATED* state. In this case, the elements associated with *T* are removed from the corresponding process model. Tasks in the *RUNNING*, *COMPLETED*, or *SKIPPED* state may not be deleted (it should not be allowed to delete a task or to change its attributes if it is already completed).

4.2 Process instance level changes

A feasibility rule at the process instance level can be triggered to control the changes at the process instance level. We instantiate process graphs, where the set of nodes can be either activities, events or gateways. The set of sequence flows (edges) connect the nodes. Let us consider *status* as an attribute assigned to each node *N* and each instance *I* to describe its current status (change-trace). The Algorithm 1 describes such a rule for the sake of illustration.

Algorithm 1. Deletion of process fragment.

```
on Ix is < deleted >
if Ix ∈ S then
    I ← Inst(Ix);
    /* Verification of corresponding instances */
    if I ∈ {Not_Activated, Activated} then
    /* the change can be applied */
        Status(Ix, "deleted");
        Mark(Ix, GREEN);
   else
       /* the change cannot be applied*/
       print("the change cannot be immediately
applied to the "+ Ix +" instance");
   end if
end if
```

4.3 Impact analysis rules

When the change is possible, impact analysis rules (analyze the impact propagation) are triggered, such as described in the Algorithm 2.

As shown in the Algorithm 2, the FO_x and other relevant control-flows are marked. The D_pO set return both succeeding activities and the corresponding routing relationships of the given activity and returns succeeding activities (if multiple activities depend on concerned activity) in respect to the different routing types: Sequential, AND-Split, AND-Join, OR-Split, OR-Join, XOR-Split, XOR-Join.

The D_pI set return both preceding activities and the corresponding routings between preceding activities and the given activity, respectively (if the concerned activity depends on multiple activities.). All returned activities and corresponding routing relationships (control-flows) are also marked to express the depth of change impact, such as shown in Fig. 3, with the help of an example.

Algorithm 2. Change Impact Analysis (Activity dependency).

```
on Rule_06 is < called >
if Status(FOx)== "added" || "deleted" || "modified"
then
    Mark(FOx, BLUE);
    Mark(FC ∈ {Ωi(FOx) ∪ Ωo(FOx)}, BLUE);
    /* Dpo gets successively each succeeding ac-
tivity depends on FOx and the corresponding rout-
ing relationships in N */
    for ai ∈ N(i = 1,...,n) do
         if ai → FOx then   /* ai depend on FOx */
              Dpo ← Dpo ∪ {ai}
        end if
    end for
   for ai ∈ Dpo(i = 1,...,n) do
       Mark(ai);
       Mark(FC ∈ {Ωi(ai) ∪ Ωo(ai)});
   end for
   /* Dpi gets successively each preceding activ-
ity which FOx depends on and
 the corresponding routing relation-ships in N */
  for ai ∈ N(i = 1,...,n) do
         if FOx → ai then   /* FOx depend on ai */
             Dpi ← Dpi ∪ {ai}
         end if
     end for
     for ai ∈ Dpi(i = 1,...,n) do
         Mark(ai);
         Mark(FC ∈ Ωi(ai) ∪ Ωo(ai));
     end for
end if
```

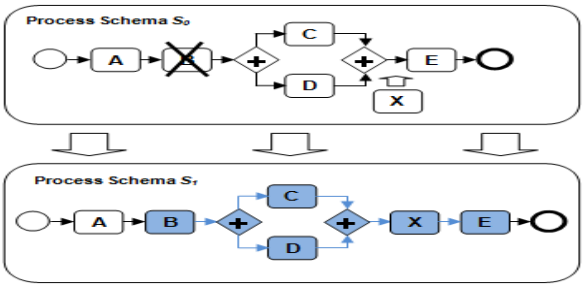

Fig. 3. Deletion of activity 'B' and addition of activity 'X'.

5 Prototype of validation

The proposed approach has been validated with the help of plug-ins development in Eclipse[2] integrated development environment. Among others, we have been developing a BPMN Change Propagation Analyzer plug-in to extend the functionality of BPM modeling for traceability of impact propagation for changing business processes. The plugin is composed of the management of meta-information of business processes and a rule-base allowing the implementation of the rules developed in context of analyzing the change integration and impact propagation.

The rule-base is implemented using the Drools[3] object-oriented rule engine in integration with Java[4]. This rule-engine allows the management of business process change impact propagation rules. Indeed, Drools is a business rules management platform that offers an integrated rule definition and execution workshop. It also allows the definition and execution of the workflow as well as the management of events. The set of impact propagation rules is interactively called when handling BPMN template elements (add, delete, or modify).

6 Conclusion and perspectives

In this paper, we propose to analyze the change impact by exploiting the dependency relationships between BPM elements. In this respect, we focus on activity, data, and role dependencies among business processes. The approach is based on graph reachability with the help of a rule-based framework. The feasibility rules and the change impact analysis rules are the two major categories of rules in this regard. These can effectively determine an *a priori* feasibility and analysis of process changes either at process type level or process instance level.

The approach has been validated with the help of set of plugins which are developed for Eclipse IDE. The continuing work aims to analyze the change impact propagation on the multiple dependency relationships, which include actors, resources, events, control data, and applications, etc. in the business process on both at the process type level, and the process instance level. The rule-based approach may provide an assistance to assess the feasibility and change impact analysis for better business process management.

References

1. M. Bouneffa and A. Ahmad, “Change management of bpm-based software applications,” in Proceedings of the 15th International Conference on Enterprise Information Systems, Angers, France. SCITEPRESS, July 2013.
2. M. Dumas, M. L. Rosa, J. Mendling, and H. A. Reijers, Fundamentals of Business Process Management. Springer Publishing Company, Incorporated, 2013.

[2] The Eclipse Foundation --- IDE and tools - https://www.eclipse.org/
[3] Drools --- Business Rules Management System - https://www.drools.org/
[4] Java - https://www.java.com/

3. J. Recker and J. Mendling, “The state of the art of business process management research as published in the bpm conference,” Business & Information Systems Engineering, vol. 58, no. 1, pp. 55–72, 2016.
4. H. Reijers, “A comprehensive approach to flexibility in workflow management systems,” in Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2006) 26-28 June 2006, Machester, United King- dom. IEEE Computer Society, 2006, pp. 271–271.
5. H. Reijers and W. van der Aalst, “The effectiveness of workflow management systems: Predictions and lessons learned,” International Journal of Information Management, vol. 25 Issue 5, pp. 458–472, 2005.
6. O. Kherbouche, A. Ahmad, and H. Basson, “Detecting structural errors in bpmn process models,” in Proceedings of the 15th IEEE International Multitopic Conference (INMIC), Islamabad, Punjab, Pakistan. IEEE Computer Society, 2012, pp. 425–431.
7. A. Awad, G. Decker, and M. Weske, “Efficient compliance checking us- ing bpmn-q and temporal logic,” in Proceedings of the 6th International Conference on Business Process Management. Springer Verlag, 2008, pp. 326–341.
8. L. Snchez-Gonzlez, F. Ruiz, F. Garca, and M. Piattini, “Improving qual- ity of business process models,” in Proceedings of the 6th International Conference, ENASE 2011, Beijing, China, vol. 275, 2013. Springer Verlag, 2013, pp. 130–144.
9. A. Ahmad, H. Basson, and M. Bouneffa, “For a better assessment of business process quality,” DEStech Transactions on Computer Science and Engineering, no. iteee, 2019.
10. M. Minor, D. Schmalen, A. Koldehoff, and R. Bergmann, “Structural adaptation of workflows supported by a suspension mechanism and by case-based reasoning,” in Proceedings of the 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises Cover. IEEE Computer Society, 2007, pp. 370–374.
11. M. Reichert, P. Dadam, and T. Bauer, “Dealing with forward and backward jumps in workflow management systems,” Software and System Modeling, vol. 1 Issue 2, pp. 37–58, 2003.
12. P. Sun and C. Jiang, “Analysis of workflow dynamic changes based on petri net,” Information and Software Technology, vol. 51 Issue 2, pp. 284–292, 2009.
13. S. Rinderle, M. Reichert, and P. Dadam, “Correctness criteria for dynamic changes in workflow systems a survey,” Data and Knowledge Engineering, vol. 50 Issue 1, pp. 9–34, 2004.
14. B.Weber, M. Reichert, and S. Rinderle, “Change patterns and change support features enhancing flexibility in process-aware information systems,” Data and Knowledge Engineering, vol. 66 Issue 3, pp. 438– 466, 2008.
15. W. van der Aalst and A. H. M. ter Hofstede, “Yawl: Yet another workflow language,” Information Systems, vol. 30 Issue 4, pp. 245–275, 2005.
16. M. L. Rosa, W. M. Van Der Aalst, M. Dumas, and F. P. Milani, “Business process variability modeling: A survey,” ACM Computing Surveys (CSUR), vol. 50, no. 1, p. 2, 2017.
17. J. Mendling, I. Weber, W. V. D. Aalst, J. V. Brocke, C. Cabanillas, F. Daniel, S. Debois, C. D. Ciccio, M. Dumas, S. Dustdar et al., “Blockchains for business process management-challenges and opportu- nities,” ACM Transactions on Management Information Systems (TMIS), vol. 9, no. 1, p. 4, 2018.
18. Y. Huang, J. Huang, B. Wu, and J. Chen, “Modeling and analysis of data dependencies in business process for data-intensive services,” China Communications, vol. 14, no. 10, pp. 151–163, 2017.
19. W. Dai and H. D. Covvey, “Query-based approach to workflow process dependency analysis,” School of Computer Science and the Waterloo Institute for Health Informatics Research,Waterloo,Ontario,Canada, Tech. Rep., 2005.