Analyzing the ripple effects of change in business process models

Oussama Mohammed Kherbouche[‡], Adeel Ahmad[†], Mourad Bouneffa[‡], Henri Basson[‡]
[‡]Université Lille Nord de France
Laboratoire d'Informatique, Signal et Image de la Côte d'Opale
BP-719 62228 CALAIS Cedex France
[†]DataSAZ Solutions
M-22, Mezzanine Floor, Mid City Plaza, Murree Road
46300 Rawalpindi, Pakistan
E-mail: {kherbouche, ahmad, bouneffa, basson}@lisic.univ-littoral.fr

***Abstract*—Change management is a critical task to control the side effects of a modification during the business process evolution. The evolution of business processes is an essential activity for the companies to better fulfill the requirements of their customers and different stakeholders. In this respect, the enterprises should adopt an effective mechanism in order to achieve the flexible business process models. It is important to identify and highlight the ripple effects of a change for minimizing their impact on other parts or entities of the system and associated services. This paper proposes a dependency-centric approach for change impact analysis. We attempt to demonstrate the change impact propagation in business process models by detecting and analyzing the interdependencies among all parts of business processes along with associated services. It can support the maintenance and evolution of business process models. The major objective is to help the modelers and business experts to assess the associated risk of intended changes and estimate the effort required for their accomplishments.**

Keywords—Business process model; change impact analysis; side effects; ripple effects; dependency relationships.

I. INTRODUCTION

The dynamic adaptability of a business process (BP) is an essential requirement for companies to cope with increasing rapid changes. However, without proper control, changes in business process models [1, 2] may generate the structural, functional, or qualitative side effects. These may include deadlocks, infinite executions, multiple endings [3, 4] or semantic conflicts [5]. Others may refer to the non-compliance with regulations [6] and the associated qualitative inconsistencies such as the degradation of service quality (response time, security, message size, etc.), or deterioration of global quality of the business process [7]. In this respect, the change management is an important subject in the life cycle of any business process and it has been actively persuaded, by the research community, for the last decade [8, 9]. The main objective of this research is to identify the potential effects of a change and to estimate the needs to accomplish a change.

In the literature, several approaches and paradigms [10-16] deal with the evolution of business processes and propose different strategies for process-instance migration during change incorporation. In [10], the author suggests a flexible modeling and execution of workflow activities based on a meta-model of business. This approach supports dynamic changes such as adding or deleting activities, but requires that the activity is not in the running state while incorporating the change. Casati *et al*. [11] presents a workflow modification language (WFML) to support the issue of migration of running instances when their respective schema is changed. It introduces the formal criteria to determine which running instances can be safely migrated to the new version. In the same way, Zhao and Liu [12] propose version management for business process schema evolution by representing different business process schema evolutions and the dependencies between them. ADEPT-flex [13] is a graph based workflow model for the integration of dynamic changes even during the execution instances of the model without losing control and structural coherence. Another approach proposed by [14] is based on the use of Petri nets to calculate the minimal region affected by the changes. The authors in [15] propose a combination of a set of patterns of change and the seven characteristics of change management. YAWL [16] is an initiative based on formal foundations that shows significant promise in the support of a number of distinct flexibility approaches.

Nonetheless, despite innovative works proposed by the BP community in the literature aimed to deal with the dynamic change management in business processes, still a lack is observed for the change impact analysis to prevent side effects [17] and/or to estimate the ripple effects [18] of concerned change.

In this paper, we propose an approach, with a differing focus as compared to existing work in the literature, which aims to increase awareness of modelers following a change in the business process models by predicting and assessing the impact of changes in a static fashion during design time. It addresses the problem in the upstream at the process type level (during design time), and not in the downstream at the instance level (during runtime). It requires an *a priori* analysis of change impact propagation in business processes through dependency relationships analysis. The proposed approach allows not only to analyze the relationships between the changed part and the other potentially affected parts in the business process model (horizontal change impact propagation) but also to analyze the effects in the concerned associated services (vertical change impact propagation).

The paper is structured as follows: the section II discusses the importance of change impact analysis in the life-cycle of business process. The typology of the dependency relationships in the business process models and associated services is discussed in the section III. Section IV proposes a set of metrics to compute the depth of change impact propagation. Section V briefly describes the change incorporation algorithms. Later, section VI concludes our contribution.

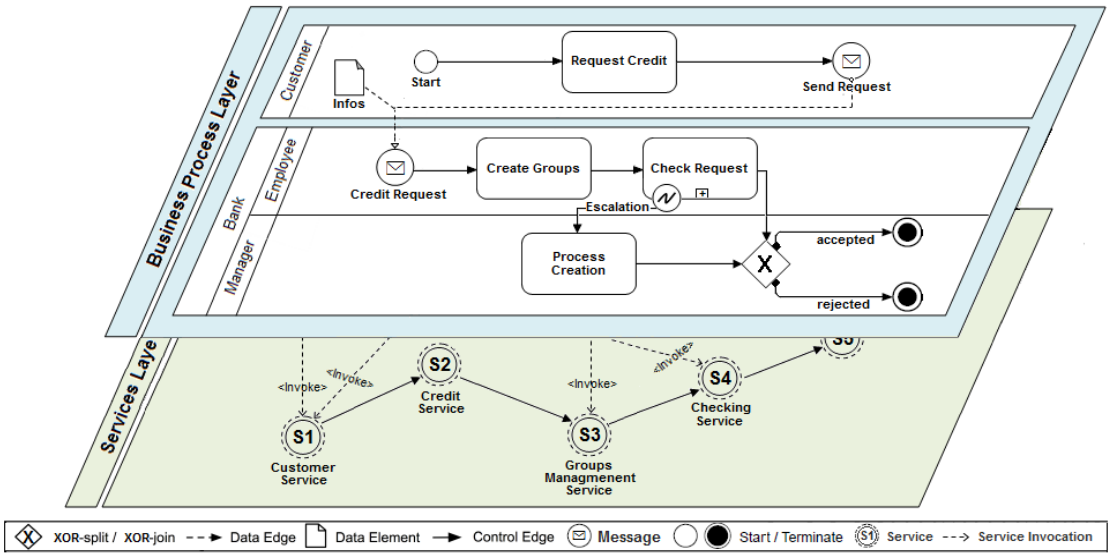

Fig. 1. Multi-layer dependency in business process

II. THE CHANGE IMPACT ANALYSIS

Causes for the changes in Business Process models (BPM) [19] can be manifold, such as a correction of one or more errors, exception handling, taking into account new legal laws, etc. It can be an evolution in the business needs (innovation), or performance improvements (optimization), etc. Unlike the first two causes which can be applied in an ad-hoc manner at process instance level (also known as instance-specific changes), the introduction of new legal laws, the innovation or the optimization of business process requires to consider the change at higher level i.e. process type level (also named process schema evolution) [20].

A change can be formally described as a difference (denoted as Δ) between the initial process schema *S* and the updated process schema *S'*, it can be quantified as follows:

$$S' = S + \Delta$$
$$\Delta = |S' - S|$$

We refer by the change impact analysis as an a priori analysis of this variant to minimize the change side effects and determine the change ripple effects in business processes. This variant can potentially generate post-change effects (in structural, functional, behavioral, logical, and qualitative aspects) [21] on partial or the whole process and which can be propagated in horizontal and/or vertical way.

The horizontal change impact propagation refers to the propagation of the change impact between the different entities belonging to the same layer i.e. the business process model layer. This is the case of the change impact propagation between the adjacent activities.

Whereas, the vertical change impact propagation refers to the propagation of the change impact between the different entities belonging to different layers i.e. the business process model layer and services layer. This is the case of the change impact propagation between a task and a web service that implements the all or a part of this task and vice versa.

III. THE DEPENDENCY RELATIONSHIPS ANALYSIS

Besides traceability analysis [22, 23], the dependency analysis play an important role in the change impact analysis process. Its main objective is to capture the existing dependency relationships in a system and to identify the entities that can be potentially impacted by a change. Indeed, like any other information system, a business process is composed of different kind of components or entities which play different roles and which also interact with each other in multiple aspects, either directly or indirectly. These interactions may refer to dependency relationships.

In order to analyze of business process dependency, it is necessary to identify and classify the primary dependency relationships.

A. Taxonomy of dependency in the business process models

We can summarize four major types of dependency which can play an important role in the change impact analysis of the business process models.

1) Structural dependency: it refers to the syntactic dependency between two entities, e.g. a change applied on an *activity* in the business process model can have a structural impact on adjacent activities.

2) Semantic dependency: it refers to the semantic relation between two entities. A change in one entity (e.g. *gateway*) can cause a change in the semantic meaning or the interpretation of the dependent entities.

3) Direct Dependency: it highlights a contol-flow ($F \subseteq n_1 \times n_2$) between two adjacent entities (n_1, n_2).

4) Indirect dependency: it is a dependency of an entity on another by a transitive or intermediate relationship. It highlights a set of intermediate control-flow that may exist between two entities. i.e. if there exist a set of direct dependency relationships between activities B_1 R B_2 and B_2 R B_n then it implies B_1 R^+ B_n.

We attempt an exhaustive analysis of the dependency relationships among the different entities of business process layers by providing a multi-dimensional dependency model that includes not only the activity and data dependency but also the different dependency relationships between business process model layer and services layer (as shown in Fig. 1).

B. Intra-layer dependency relationships

The intra-layer business process dependency encompasses the followings:

1) Activity dependency (routing): The activity dependency or routing dependency describes the activity and process execution order through the control-flows (i.e. *sequence flow* and *message flow*). These execution orders are based on technical requirements and business regulations. The activity dependency defines not only the execution order but also the semantics associated with this order. For example, an *XOR-Join* routing of the three activities *A, B* and *C* such that *A* or *B* must execute before *C*. We can further distinguish two kinds of routing relationships [24].

- Intra-process dependency: The intra-process dependency refers to routing relationships between neighboring activities within the same process, as shown in Fig. 1 (within the bank process).
- Inter-process dependency: It is concretely represented by the routing relationship between activities in the different processes (e.g. messages exchange). As shown in the Fig. 1 (between the customer and bank process).

The activity dependency can be formally defined as: $\mathcal{d}a = (\mathcal{D}_p, \Omega)$ over a set of activities $A = \{a_1 \ldots a_n\}$ and a set of control-flows $T = \{t_1 \ldots t_n\}$, where:

$$\mathcal{D}_p = \mathcal{D}_p i\,(a) \cup \mathcal{D}_p o\,(a) \text{ whereas } a \in A.$$

$\mathcal{D}_p o\,(a)$ is a set of all succeeding activities $a_i \in A$ (denoted as: $a_i \rightarrow a$) where the executions are dependent on activity a. The relationship can be one-to-many i.e., multiple activities depend on one activity. In the same way, $\mathcal{D}_p i\,(a)$ 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 may be a many-to-one, i.e., one activity depends on multiple activities.

The set of control flows can be formaly shown as:

$$\Omega = \Omega i \cup \Omega o$$

The Ωi is a set of control-flows, $ti \in T$, connecting each activity $a_i \in \mathcal{D}_p i(a)$ to a, i.e. all incoming arcs $(\mathcal{D}_p i(a), a)$ of a. While the Ωo is a set of control-flows, $ti \in T$, connecting a to each activity $a_i \in \mathcal{D}_p o(a)$, i.e. all outgoing arcs $(\mathcal{D}_p o(a), a)$ of a.

2) Data dependency : The data dependency refers to the common resources or data related to multiple activities. There exist three major types of data dependencies [25]:

- Flow dependencies: It emerges whenever one activity produces a resource or data that is used by another activity.
- Sharing dependencies: It occurs whenever multiple activities use the same resource or data.
- Fit dependencies: It arises when multiple activities collectively produce a single resource or data.

We can formalize a set of all data transferred between activities [13] as:

$$D = \{d_1, d_2 \ldots dn\}$$

Every activity $a \in A$ has input and output parameters, denoted as *InPARs(a)* and *OutPARs(a)* parameters, respectively. The symbol *dc* represents a data connection as:

$$dc = \{d, a, par, mode\}$$

Where $d \in D$, $a \in A$, $par \in InPARS(a) \cup OutPARs(a)$, and $mode \in \{read, write\}$. The set of all data connections can be represented as:

$$DC = \{dc_1, dc_2, \ldots, dc_n\}$$

An activity a_i depends on another activity a_j *(denoted as:* $a_i \xrightarrow{(D)} a_j$*)* iff: $\exists\, dcx, dcy \in DC$, Such that:

$$dcx = (d, a_j, pars, write)$$
$$dcy = (d, a_i, part, read)$$

Where $d \in D$, $part \in InPARs(ai)$, $pars \in OutPARs(aj)$ and a_j precedes a_i in process schema.

C. Inter-layer dependency relationships

Like the business process layer, the services layers contains a set of services which interact with each other by direct or indirect relationships and which are also subjected to consistent evolution[26]. But there exist beside, a coupling relation between services and business processes as shown in the Fig. 1 which we have called inter-layer dependency relationship.

Indeed, the correlation, input and output of Web services are related through process orchestration which uses process-based service standards such as WS-BPEL [27]. Therefore, a business process may support multiple services. This implies that whenever a change occurs in the business process, the change may affect the services that are associated with this business process and vise versa. This can be justified by the fact that an activity in the business process model can interact with a service in uni or bidirectional way via the invocation of one of its operations.

A service is generally defined as a set of operations $o_i \in O = \{o_1, \ldots, o_n\}$ which is associated with a set of messages and the invocation relations $T \subseteq O \times O$ associated with the operations.

We can formally define the dependency relationship between the business process layer and service layer as: $\mathcal{d}i = (A_s, r)$ over a set of activities $A = \{a_1 \ldots a_n\}$ and a set of web services $S = \{s_1 \ldots s_n\}$, where:

A_s is a set of activities that invoke the same service $s_i \in S$ (denoted as : $a_i \rightsquigarrow s_i$) and $r \subseteq A \times S$ is the set of edges connecting nodes that represent the dependency

relationships between each activity $a_i \in A_s$ and invoked service $s_i \in S$.

IV. CHANGE IMPACT METRICS DESCRIPTION

The analysis of the change ripple effects should not be limited to know how many and which other part of process can be impacted in the business process. But it should also analyze from where the change impact begins and where it ends i.e. compute the impact propagation depth while distinguishing between the impacted entities in term of degree of the impact i.e. impact power.

To deal with this problem, we propose to assign a numeric weight to each dependency relationship in the corresponding business process layer and the dependency relationship between businesses process layer and services layer. This assigned value (*Impact Weight Factor*) intends to compute the depth of impact and thus the degree of the impact for each impacted entity.

In fact, the *Impact Weight Factor* represents a numeric value which can be used to measure the change impact propagation by expressing the impact level of one entity to another. It is calculated on the basis of a set of metrics, as follows:

$$IWF\ (a_i) = \sum (P\ (a_i) + E\ (a_i) + F\ (a_i) + ND\ (a_i))\ /TDR$$

Where, *TDR* represents the total number of dependency relationships in the model. Whereas the different metrics are described as below:

A. The priority metric

The priority metric, denoted as *P,* represents the priority of a given activity $a_i \in A$ in the execution flow of the business process model. It is fixed by the modelers and business experts during the design-time. It can be computed from the corresponding value to one of the three values: High, Medium and Low i.e. $P(A) = 0.75 \Leftrightarrow 75/100$.

B. The execution frequencies metric

The execution frequencies metric of a given activity in each execution path of business process instance I_x where $x \in \{1 \ldots n\}$ can be calculated from the process execution logs. Indeed, the business process execution logs could be mined to discover the number of times each execution path is run for an activity. For example, if we observe that the execution path that lead to the activity *A* and *B* is executed rather than the path leading to the activity *C* in the different instances, it implies that the *E(A)* and *E(B)* is greater than *E(C)*.

C. The invocation frequencies metric

This metric refers to the invocation frequencies denoted as *F* of a given activity in the business process model. For example, if the activity *B* is called in every possible execution path of activity *A*, then the likelihood of *A* being impacted, by a change in *B,* becomes high. However, if *B* is called in only one among many possible execution paths inside *A*, then the likelihood of *A* being impacted, by a change in *B,* becomes much lower.

D. The nested depth metric

*T*he nested depth metric *ND* is the position of each activity in the execution path of the business process model. In other terms, the activity that occurs formerly in the path of business process model receives a higher coefficient, while the activity that occurs later in the path of business process model receives a lower coefficient.

The degree of the impact for each impacted entity by a change can be calculated as a sum of all the *Impact Weight Factors* from the changed entity to impacted entities. In other words, the entities involved in dependency relationship with the changed entity are likely to be impacted with their inclusion in the path which connects them i.e. we investigate all its possible execution paths and we assign an impact value for each path.

Therefore, we say that a path can be marked as red when the degree of the impact is high e.g. between 15 and 20. We say that a path can be marked as orange when the degree of the impact is medium e.g. between 10 and 15 and that a path can be marked as green when the degree of the impact is low e.g. between 1 and 10. These threshold values can be defined by modelers and business expert based on empirical studies.

V. CHANGE IMPACT PROPAGATION ASSESSMENT

In order to provide a generic approach for change impact analysis in the business process model and associated service, we propose a high level reasoning using an abstract notation i.e. we do not presume any particular process modeling approach, but simply take in to account the basic elements of a process. A business process is expressed in an abstract way (e.g. using graph based formalism).

A. *Definition 1 (Business process):* Formally, a business process (*Gp*) is a function *Gp (N, Type, F, Status)* where:

- $N = \{n_1, n_2, \ldots, n_n\}$ is a finite set of nodes which can be partitioned into events, activities, and gateways, represented respectively as *E, A, G*).
- Type: $N \rightarrow$ {Activity, Event, AND-join/split, XOR-join/split, OR-join/split} is a function that assigns a type to each node.
- $F \subseteq N \times N$ (edges in the graph) represents the sequence flow relation between the nodes. Such that *{(n, n') ∈ F}* denote, respectively, the direct predecessors and successors of a node $n \in N$.
- S is the set of services invoked by the process.
- D is a set of data.
- Status $\rightarrow$ {init, added, deleted, modified} is the current status (change-trace) of node $n \in N$ in process model.

B. *Definition 2 (Activity):* An activity $a_i \in A$ is can be defined on order *(Input, Output, s)* where $Input \subset D$ is the set of a_i inputs, $Output \subset D$ is the set of a_i outputs and $s \in S$ is the service invoked by a_i.

The algorithm 1 presents the rule of change impact propagation through activity dependency relationships. It is triggered when a modeler attempts to add, modify or remove a flow-object FO_x from a process at process type level. The FO_x and these control-flows are marked. The *Dpo* set return both succeeding activities and the corresponding routing relationships between a given activity and returns succeeding activities (multiple activities depend on concerned activity).

The set *Dpi* return both preceding activities and the corresponding routings between preceding activities and a given activity, All returned activities and corresponding routing relationships (control-flows) are also marked to express the depth of change impact which is calculated on the basis of *Impact Weight Factor*.

Algorithm 1: *Intra-layer dependency relationships Analysis (Activity dependency)*

```
Input: N // set of nodes
Init : Dpo ← ∅, Dpi ← ∅
Begin
 If Status (FOx) = ="added" ||"deleted" ||"modified" then
   mark(FOx);
   mark (F ∈ {Ωi(FOx) ∪ Ωo(FOx)});
 /* Dpo gets successively each succeeding activities
   depends on FOx and the corresponding routing
   relationships in N */
 for all ai ∈ N (i = 1,…, n) do
    /* ai depend on FOx */
    If ai → FOx then
      Dpo ← Dpo U {ai}
      IWF← CalcIWF(ai);
      pathImp ← calcPathImpact(execPath(ai),IWF);
      putInMap(ai, pathImp) ;
    end if
 end for
 /* Dpi gets successively each preceding activities
    which FOx depends on and the corresponding
    routing relationships in N */
 for all ai ∈ N (i = 1,…, n) do
   /* FOx depend on ai */
   If FOx → ai then
      Dpi ← Dpi U {ai}
      IWF← CalcIWF(ai);
      pathImp ← calcPathImpact(execPath(ai),IWF);
      putInMap(ai, pathImp) ;
   end if
 end for
 for all ai ∈ Dpo U Dpi (i = 1,…,n) do
   ImpactPow ← getFromMap(ai) ;
   clr ← getColorPathImpact(ImpactPow);
   mark(ai, clr) ;
   mark(F ∈ {Ωi(ai) ∪ Ωo(ai)}, clr)
 end for
end if
End
```

The second rule as described in Algorithm 2 is also triggered when the change at the process type level occurs. The *OutputDs* represent a set of data produced by FO_x activity and *AllOutDep* is a set of activities that depend on a specific output data element *D* produced by activity FO_x. The *InputDs* is a set of data used by FO_x activity and *AllInDep* is a set of activities where the output data is the input data of an activity FO_x.

Algorithm 2: *Intra-layer dependency relationships Analysis (Data dependency)*

```
Input: N // set of nodes
Init : OutputDs← ∅, AllOutDep ← ∅, InputDs← ∅,
       AllInDep ←∅
Begin
 If Status (FOx) = ="added" ||"deleted" ||"modified" then
   mark(FOx);
   mark (F ∈ {Ωi(FOx) ∪ Ωo(FOx)});
   for all Di ∈OutPARs(FOx) (i = 1,…, n) do
      OutputDs← OutputDs U {Di}
   end for
   for all Di ∈OutputDs(i = 1,…, n) do
     mark(Di);
     mark(F∈ {Ωi(Di) ∪ Ωo(Di)});
    /* AllOutDep gets successively each activities
      which depend on output data of the given activity
      FOx in N */
      For all aj∈ N( j = 1,…,n) do
       /* aj depend on D produced by FOx */
        If aj -(D)-> FOx then
          AllOutDep ← AllOutDep ∪ {ai}
          IWF← CalcIWF(ai);
          pathImp ← calcPathImpact(execPath(ai),IWF);
          putInMap(ai, pathImp) ;
        end if
      end for
      end for
      for all Di∈ InPARs(FOx)(i = 1,…, n) do
        InputDs ← InputDs ∪ {Di}
      end for
      for all Di ∈ InputDs(i = 1,…,n) do
        mark(Di);
        mark(F∈ {Ωi(Di) ∪ Ωo(Di)});
       /* AllInDep gets successively each Activities
          where the output data is the input data of given
          activity FOx in N */
       for all aj ∈ N( j = 1,…,n) do
         /* FOx depend on D produced by aj */
         If FOx -(D)-> aj then
           AllInDep ← AllInDep ∪ {aj}
           IWF← CalcIWF(aj);
           pathImp ← calcPathImpact(execPath(aj),IWF);
           putInMap(aij, pathImp) ;
         end if
       end for
       for all ak ∈ ALLInDep U ALLOutDep(k = 1,…, n) do
        ImpactPow ← getFromMap(ak) ;
        clr ← getColorPathImpact (ImpactPow);
        mark(ak, clr) ;
        mark(F∈ {Ωi(ak) ∪ Ωo(ak)} , clr);
      end for
    end for
  end if
End
```

The third rule as described in Algorithm 3 analyzes the impact propagation in vertical way. The *Dps* represent a set of all services invoked by concerned activity. All returned services and corresponding invocation relationships (service invocation) are also marked to express the depth of change impact.

Algorithm 3: *Inter-layer dependency relationships analysis*

```
Input: N // set of nodes
Init : , Dps ← ∅
 Begin
  If Status (FOx) = ="added" ||"deleted" ||"modified" then
    mark(FOx);
    mark (F ∈ {Ωi(FOx) ∪ Ωo(FOx)});
  /* Dps gets each services invoked by FOx and the corresponding
     routing relationships */
   for all si ∈ S (i = 1,…, n) do
     /* si is invoked by FOx */
```

```
      If FOx ⇝ si then
        Dps ← Dps U {si}
        IWF← CalcIWF(si);
        pathImp ← calcPathImpact(invokepath(si),IWF);
        putInMap(si, pathImp) ;
      end if
    end for
   for all si → Dps (i = 1,…,n) do
      ImpactPow ← getFromMap(si) ;
      clr ← getColorPathImpact(ImpactPow);
      mark(si,clr) ;
      mark(F ∈ {r(FOx,si)},clr)
   end for
end if
End
```

VI. CONCLUSION

The change management in business process can help to analyze change impact and the generated ripple effects. In this paper, we propose an approach for an *a priori* analysis of change impact propagation in business processes by exploiting the dependency relationship between the modified part and other potentially affected parts by concerned changes. In this respect, we focus on the dependency relationships analysis within the business process model and among business process models and the services that implement them. We use a set of metrics to calculate the significant depth of change impact propagation depth while distinguishing the impacted entities in term of degree of the impact. The change impact propagation is assessed on the basis of graph reachability. The continuing work aims both to analyze the change impact propagation on other dependency relationships (which include actors, resources, events, control data, and applications, etc.) in the business process and to analyze other layers which may likely be impacted by a change i.e. components layer, data layer, etc.

REFERENCES

[1] R.Lu "Constraint-Based Flexible Business Process Management," in School of Information Technology and Electrical Engineering, University of Queensland, 2008.

[2] W. van der Aalst, and al, "Business Process Management: A Survey," in Proceedings of Conference on (BPM 2003), Eindhoven, Netherlands, pp 1-12, 2003.

[3] O.M Kherbouche, A.Ahmad, H. Basson, "Detecting structural errors in BPMN process models," in Proceedings of the 15th International Multitopic Conference (INMIC), Islamabad, Punjab, Pakistan, pp. 425- 431, IEEE Computer Society, 2012.

[4] O.M Kherbouche, A.Ahmad, H. Basson, " Using model checking to control the structural errors in BPMN models," in Proceedings of the 7th International Conference on Research Challenges in Information Science (RCIS), Paris, France, IEEE Computer Society, 2013.

[5] S. Rinderle, M. Reichert, and P. Dadam, "On dealing with semantically conflicting business process changes," Technical Report UIB-2003-04, University of Ulm, Computer Science Faculty, June 2003.

[6] A. Awad, G. Decker, M. Weske, "Efficient Compliance Checking Using BPMN-Q and Temporal Logic," in Proceedings of the 6th International Conference on Business Process Management, pp. 326-341, 2008.

[7] L.Sánchez-González, F. Ruiz, F. García, M. Piattini, "Improving Quality of Business Process Models," in Proceedings of the 6th International Conference, ENASE 2011, Beijing, China, pp. 275-144, 2011.

[8] B. A. Rajabi, S. P. Lee, "Modeling and Analysis of Change Management in Dynamic Business Process," in International Journal of Computer and Electrical Engineering, pp. 181-189, 2010.

[9] H.A. Reijers, "Workflow Flexibility: The Forlorn Promise," in Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2006), 26-28 June 2006, Manchester, United Kingdom, pp. 271-272. IEEE Computer Society, 2006.

[10] M. Weske, "Flexible modeling and execution of workflow activities," in Proceedings of the Thirty-First Hawaii International Conference, vol. 7, Jan. 6-9, pp. 713-722, 1998.

[11] F. Casati, S. Ceri, B. Pernici and G. Pozzi, "Workflow Evolution," Published in Data and Knowledge Engineering, pp. 438-455, 1996.

[12] X. Zhao, C. Liu, "Version management for business process schema evolution," Published in Information Systems, Volume 38, Issue 8, pp. 1046–1069, 2013.

[13] M. Reichert and P. Dadam, "ADEPTflex - Supporting Dynamic Changes of Workflows without Losing Control," Journal of Intelligent Information Systems, Special Issue on Workflow Management, Vol. 10, pp. 93-129, 1998.

[14] P. Sun and C. Jiang, "Analysis of workflow dynamic changes based on Petri net," in Information and Software Technology, Volume 51, Issue 2, pp. 284-292 February 2008.

[15] B.Weber, M. Reichert, S. Rinderle, "Change patterns and change support features – Enhancing flexibility in process-aware information systems," in Proceedings of the 19th international conference on Advanced information systems engineering, pp. 574-588, 2008.

[16] W.M.P. van der Aalst and A.H.M. ter Hofstede, "YAWL: Yet Another Workflow Language, " Published in Information Systems, 30(4) pp.245-275, 2005.

[17] R. S. Arnold and S. A. Bohner, "Impact Analysis - Towards A Framework for Comparison," in Proceedings of the Conference on Software Maintenance, Los Alamitos, CA, pp. 292-301, 1993.

[18] S. A. Bohner, "A Graph Traceability Approach for Software Change Impact Analysis," Ph.D. Dissertation, George Mason University, Fairfax VA, 1995.

[19] S. Rinderle, M. Reichert, and P. Dadam, "Correctness criteria for dynamic changes in workflow systems - a survey," Published in Data Knowledge., vol. 50, no. 1, pp. 9–34, 2004.

[20] S. Nurcan, "A Survey on the Flexibility Requirements Related to Business Processes and Modeling Artifacts," in Proceedings of the 41st Hawaii International Conference on System Sciences HICSS'2008, pp. 378-378, 2008

[21] O.M Kherbouche, M. Bouneffa, A.Ahmad, H. Basson, " Analyse a priori de l'impact du changement des processus métiers," in Proceedings of the 31th INFormatique des ORganisations et Systèmes d'Information et de Décision (INFORSID), Paris, France, 29-31 May 2013.

[22] A. Marcus, J. I. Maletic, "Recovering Documentation to Source Code Traceability Links using Latent Semantic Indexing," in Proceedings of the 25th International Conference on Software Engineering, pp. 125- 135, 2003.

[23] A. V. Knethen, "A Trace Model for System Requirements Changes on Embedded Systems," in Proceedings of the 4th International Workshop on Principles of Software Evolution, pp. 17-26, 2001.

[24] G. Grossmann, G. Schrefl, M. Stumptner, "Modelling inter-process dependencies with high-level business process modelling languages," in Proceeding of the Asia-Pacific Conference on Conceptual Modelling (APCCM). pp. 89–102, 2008.

[25] T. W.Malone, "Tools for inventing organizations: Toward a handbook of organizational processes", Published in Management Science 45(3) pp. 425-443, 1999.

[26] W. Shuying, L.F. Capretz, "A Dependency Impact Analysis Model for Web Services Evolution", in Proceeding of IEEE International Conference on Web Services, ICWS 2009, Los Angeles, CA, pp. 359 - 365 2009.

[27] WSBPEL Web Services Business Process Execution Language Version 2.0, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html