IET Software

Research Article

IET Journals
The Institution of Engineering and Technology

Data-driven business process similarity

ISSN 1751-8806
Received on 2nd October 2016
Revised 10th March 2017
Accepted on 11th April 2017
doi: 10.1049/iet-sen.2016.0256
www.ietdl.org

Mohammad Javad Amiri[1] ✉, *Mahnaz Koupaee*[1]

[1]*Department of Computer Science, University of California Santa Barbara, Santa Barbara, CA, USA*
✉ *E-mail: amiri@cs.ucsb.edu*

Abstract: Although measuring the similarity of business processes based on activity labels, structural and behavioural factors can be effective, defining inexact and incomplete labels and the existence of multiple labels for similar activities cause challenges for determining similar processes. Recent attempts to consider data in business process management and the support of data modelling in business process standards have led to the creation of multiple business models with data access. In this study, a method considering data for measuring business process similarity is presented in which first the similarity of activities is measured according to their structures and behaviours in a process and also their data access. Then based on the similarity of activities, the similarity of processes is determined using the proposed algorithm.

1 Introduction

The increasing application of workflow systems in different businesses and organisations raised a lot of new issues in the field of business process management. Many large enterprises require hundreds of processes to fulfil their duties. For example, the total number of business process models in the Systems, Applications and Products (SAP) reference model or the repository of Dutch Local Governments exceeds 600 [1] and in office automation (OA) System of China Mobile Communications Corporation (CMCC) this number has been over 8000 [2] of which many are similar or even identical. While most existing approaches rely on the structure of the process and label of activities to find similar processes, in this paper, we study the role of data in process similarity measurement.

Measuring the similarity of business processes is significant in business process management domain for various reasons. First, there may be identical or similar processes being executed in different parts of an organisation; the identification and merging these processes prevent the duplication of activities. Furthermore, businesses are constantly being changed and extended. In many cases, separate small businesses unite with each other and form a unique business; identifying similar business processes can be used to reduce the cost of expanding businesses [3]. Similarity measurement can also be used by multinational enterprises to identify national branch processes that no longer comply with enterprise reference model [4]. In brief, scenarios such as organisation merging, user requirements changing, and model repository management are some applications of process similarity measurement [2].

Most of similarity measurement approaches depend on the labels of activities. To calculate the similarity of labels of different activities, schema matching [5] and ontology matching [6] techniques are used. Labels can be compared either syntactically or semantically. To compare the labels on the syntactic level, string edit distance is used in some approaches [7]. Some other methods tokenize strings into words and then compare the similarity [8]. Natural language processing techniques [9, 10] are used to compare labels on semantic level. There are several drawbacks of using labels in similarity measurement. First of all, labels may be chosen in an inexact and incomplete way that does not reflect what really a task does. In addition, there may be meaningless labels which do not convey useful information about the desired activity. Moreover, different words or synonyms may be used to describe the same activity [11]. So there may be an activity with multiple labels. Although there have been a lot of attempts to overcome the challenges caused by labels such as ontology-based techniques and Natural Language Processing (NLP), which can help reduce the negative effects of the mentioned problems, yet the accuracy of those methods are not satisfactory.

On the other hand, there has been a significant attention towards data role in business process management. In order to fulfil a business process, the various data objects are required and the result of process execution can be observed in creation or update of data. For this reason, data is considered as the core of a business process in different frameworks and methods [12–14]. Modelling data access as a part of a process is supported in modelling standards such as Business Process Model and Notation (BPMN) and organisations attempt to model data while modelling the process in order to show their processes more exactly.

There have been various advantages of modelling data dependencies in processes. First, having an integrated view of tasks and data facilitates communication with stakeholders about processes and their data manipulations. In addition, since the model contains complex data dependencies, automatic enactment of processes can be performed from the model only. Different process representations such as models showing the evolution of a data object throughout a process can also be generated automatically. Finally the integrated model of control and data flow help analysing the consistency and correctness of processes [15].

To overcome the problem of different, multilingual or meaningless labels, and due to the importance of using data, the proposed method uses the similarity of data access patterns to find similar processes that use the same data model. In fact, the problem can be defined as follows: Given a pair of business processes which have access to same data model, find the similarity of those two processes.

The proposed method uses the structural, behavioural and data access similarities as the basis of similarity measurement process. To do so, processes with their data accesses (read or write) are modelled as graphs. Graphs enable us to compare different processes modelled with different modelling languages such as BPMN and Event-driven Process Chain (EPC). Then, the mutual similarity of activities of different processes is calculated. The structural and behavioural similarity is calculated based on the position of activity in a process, the number of successor, predecessor, parallel activities and so on. The data similarity is calculated based on the similarity of access to data. By computing structural, behavioural, and data similarity, the similarity of two different processes can be calculated. First, one process is considered as the base process and for each activity the most similar activity from the other process is selected. By using an

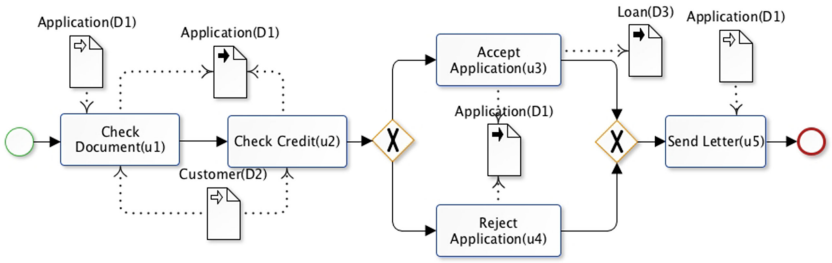

Fig. 1 *First type of loan process*

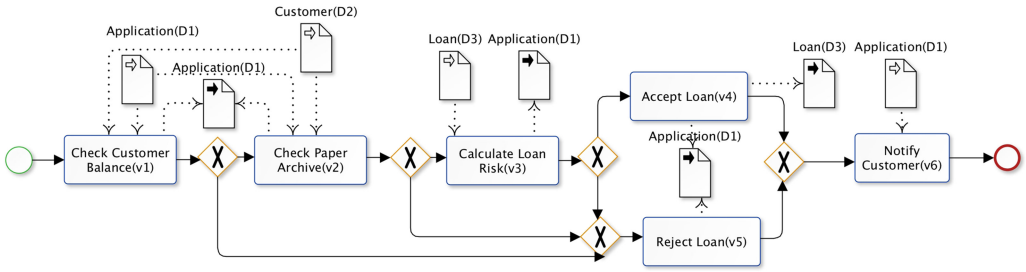

Fig. 2 *Second type of loan process*

algorithm like A*, but with higher efficiency due to prevention of extending extra nodes, the total similarity is calculated.

This paper is organised as follows. Process modelling is presented in Section 2. The focus of Section 3 is on measuring similarity between activities. Section 4 provides an algorithm to measure similarity between processes. In Section 5, an evaluation of our method is presented. Related work is discussed in Section 6, and Section 7 concludes the paper.

2 Process modelling

In this section, we introduce a formal model for business processes. The key notions include 'business process graph', 'activity vector', and 'activity vector operation'.

Business process refers to a set of related activities which is done co-ordinately to achieve a product or a business service [16]. Modelling these processes is performed with the aim of showing and understanding the execution of a task better. Also business processes are used for several purposes in domain of service computing, such as service identification [17] and service composition [18]. EPC, UML activity diagram and BPMN are the most common modelling languages. The main elements of a process include activities, events, gateways, and control flow edges [19]. As it is mentioned before, due to the importance of data, data objects and data flow edges which show the access of activities to data are added to process elements. Although all the above-mentioned languages can be used to model a business process, a graph structure to support all modelling languages is used in this paper. Mapping a process to a graph increases the generality of the proposed method and also provides the possibility for using graph similarity techniques.

Example 1: Figs. 1 and 2 show two types of loan processes. These processes are modelled using BPMN 2.0 standard. In the first process, the documents of loan applicant are checked. This information is obtained from the application and the customer objects and the result of checking is written on the customer's application. The next step is checking the customer credit and writing the results on the application. On the basis of these two steps, the bank makes a decision to accept or reject the application. The decision should be written on the application. In case of acceptance, a new record is created in the loan object. Finally, the bank informs the customer about the decision. The second process is almost similar to the first one, but it first checks the customer balance and then checks the paper archive. It also has another step to check the loan risk. In each step if the customer situation is not satisfactory the bank rejects the loan and notifies the customer.

There are several researches on mapping a business process to a graph where we use some notations from [20]. In our formal model, a 'process' is a graph with edges corresponding to control flow and data flow transitions and nodes representing start/end events, activities, gateways, or data object. We assume the existence of countably infinite, pairwise disjoint sets of U_A, U_X, U_{AND}, U_{XOR}, and U_D as activities, events, And (split/join) gateways, XOR (choice/merge) gateways, and object names, respectively. Let $U_G = U_{AND} \cup U_{XOR}$ be the set of gateways and $U = U_A \cup U_X \cup U_G$ be the set of nodes. Two classes of *events* are considered: *start* and *end* events. An *activity* represents an atomic unit of work. A *gateway* node in a process alters the current execution path. There are four kinds of frequently used gateways in process modelling standards: *choice*, *merge*, *split*, and *join*. Most of the proposed graph structures for modelling processes ignore gateways; this may lead to decrease in the accuracy of similarity measuring method because there is no difference between a parallel gateway and a choice gateway. However, the parallel execution of processes differs a lot from the execution of a chosen path. Although considering gateways can improve accuracy, merging them can cause challenges. For instance, in BPMN two similar gateways can be connected while in Petri nets two gateways are merged. In fact, showing two similar gateways separately or merging them does not make any differences in business logic; therefore in the proposed method for two consecutive gateways from the same type, only one node is used. This cannot be applied to gateways with different types. Finally, A *data* (node) shows a data object.

Definition: (Business Process Graph): A *Business Process Graph* is a tuple $G = (N, s, F, V, D, O)$ such that

- $N \subseteq U$ is a finite non-empty set of control flow nodes,
- $s \in (N \cap U_X)$ is the start node,
- $F \subseteq (N \cap U_X) - \{s\}$ is a finite set of final nodes,
- $V \subseteq (N - F) \times (N - \{s\})$ is a finite set of control flow relations such that

  i. s has one outgoing edge and no incoming edges,
  ii. each node in F has one incoming edge and no outgoing edges,
  iii. each node in $N \cap U_A$ is an activity node with one incoming and one outgoing edge, and
  iv. for each node in $N \cap U_G$, the number of incoming edges plus number of outgoing edges are at least three,
- $D \subseteq U_D$ is a finite set of data object nodes, and

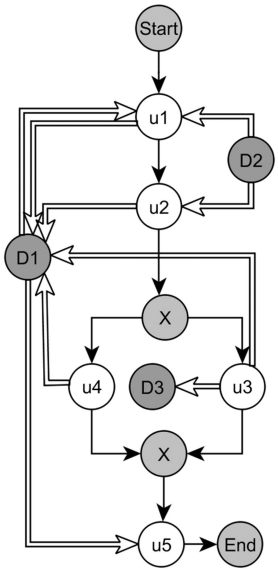

Fig. 3 *Loan process (type 1) graph*

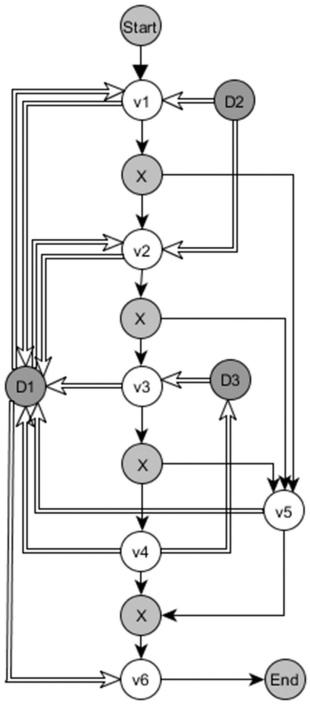

Fig. 4 *Loan process (type 2) graph*

- $O \subseteq ((N \cap U_A) \times D) \cup (D \times (N \cap U_A))$ is a finite set of data flow relations.

Different business process graphs have different sizes that can influence their similarities. The size of a graph usually is defined as the number of its nodes. For process graph, we define the size of graph G (*size(G)* or $|G|$) as the number of activities inside the process. We use graph size later on to measure similarity.

Example 2: Figs. 3 and 4 represent the business process graphs corresponding to the business process models of Figs. 1 and 2, respectively. As it can be seen, four types of nodes are specified in those graphs: start/end nodes, data object nodes, activity nodes, and gateway nodes. Also two types of edges are defined: control flow edges, which are directed solid black lines and data flow edges, which are directed double white lines. For the first process, the size is equal to 5 and the size of second one is equal to 6.

One of the most important parts of the proposed formal model is data flow relation which is used to show data accesses. To show the data accesses for each activity, an activity vector is defined. The activity vector has two elements for each data object; one represents the read access, and other one represents the write access. Also weight can be defined for each activity vector. Since in the literature the weight of write access is considered as twice as the weight of read access, to calculate the weight of an activity vector, the weight of elements that correspond to the write accesses are doubled.

Definition: (Activity Vector): Let $G = (N, s, F, V, D, O)$ be a process graph, $t \in (N \cap U_A)$ an activity(task) node in G, and $|D| = m$, $\boldsymbol{\alpha}_t = (v_1, v_2, \ldots, v_{2m})$ is an *activity vector* for activity node t where for each $i \in [1, \ldots, m]$, $d_i \in D$,

- if $(d_i \times t) \in O$ then $v_{2i-1} = 1$,
- if $(t \times d_i) \in O$ then $v_{2i} = 1$.

Let $\boldsymbol{\alpha}_t = (v_1, v_2, \ldots, v_{2m})$ be an activity vector corresponding to activity node t, *weight* of $\boldsymbol{\alpha}_t$ defines as

$$w(\alpha_t) = \sum_{k=1}^{m} (v_{2k-1} + 2(v_{2k}))$$

Example 3: Continuing with Example 2, for loan processes activities, activity vectors are defined as follows:

$$\boldsymbol{\alpha}_{u_1} = (1,1,1,0,0,0), \quad \boldsymbol{\alpha}_{u_2} = (0,1,1,0,0,0),$$
$$\boldsymbol{\alpha}_{u_3} = (0,1,0,0,0,1), \quad \boldsymbol{\alpha}_{u_4} = (0,1,0,0,0,0),$$
$$\boldsymbol{\alpha}_{u_5} = (1,0,0,0,0,0)$$
$$\boldsymbol{\alpha}_{v_1} = (1,1,1,0,0,0), \quad \boldsymbol{\alpha}_{v_2} = (1,1,1,0,0,0),$$
$$\boldsymbol{\alpha}_{v_3} = (0,1,0,0,1,0), \quad \boldsymbol{\alpha}_{v_4} = (0,1,0,0,0,1),$$
$$\boldsymbol{\alpha}_{v_5} = (0,1,0,0,0,0), \quad \boldsymbol{\alpha}_{v_6} = (1,0,0,0,0,0)$$

Representing the data accesses of activities using vectors helps us define some operations on those vectors. These operations are used in the future to define data similarity between different activities. Operations are simple logic operations such as AND, OR, and XOR. These operations perform on same elements of any two activity vectors to create a new vector. Basically, *AND* of two activity vectors shows the same data access of those activities. *OR* represents all the data that are accessed by any of those activities, and *XOR* indicates the data which is accessed by only one of those two activities. To do so we need to ensure that the size of activity vectors are equal, because of that we union the data nodes' sets add sufficient zero digits to the vectors.

Definition: (Activity Vector Operations): Let $G = (N, s, F, V, D, O)$ and $G' = (N', s', F', V', D', O')$ be two graphs and $t \in (N \cap U_A)$, $t' \in (N' \cap U_A)$ be two activity nodes in G and G', respectively. Without loss of generality $\boldsymbol{\alpha}_t = (u_1, u_2, \ldots, u_{2m})$ and $\boldsymbol{\alpha}_{t'} = (v_1, v_2, \ldots, v_{2m})$ where m is equal to $|D \cup D'|$.
Three *activity vector operations* are defined as follows:

- $\boldsymbol{\alpha}_{AND(t,t')} = (w_1, w_2, \ldots, w_{2m})$ is an activity vector, where

$$\forall k \in [1, \ldots, 2m]: w_k = u_k \wedge v_k,$$

- $\boldsymbol{\alpha}_{OR(t,t')} = (x_1, x_2, \ldots, x_{2m})$ is an activity vector, where

$$\forall k \in [1, \ldots, 2m]: x_k = u_k \vee v_k,$$

- $\boldsymbol{\alpha}_{XOR(t,t')} = (y_1, y_2, \ldots, y_{2m})$ is an activity vector, where

$$\forall k \in [1, \ldots, 2m]: y_k = u_k \oplus v_k\,.$$

For each of these three vectors, weight can be calculated as stated before.

Example 4: Continuing with Example 2, following vectors show activity vector operations and the weight of each vector on some activities:

- $\boldsymbol{\alpha}_{AND(u_2, v_2)} = (0, 1, 1, 0, 0, 0)$, where $w(\boldsymbol{\alpha}_{AND(u_2, v_2)}) = 3$,
- $\boldsymbol{\alpha}_{OR(u_2, v_4)} = (0, 1, 1, 0, 0, 1)$, where $w(\boldsymbol{\alpha}_{OR(u_2, v_4)}) = 5$,
- $\boldsymbol{\alpha}_{XOR(u_4, v_4)} = (0, 0, 0, 0, 0, 1)$, where $w(\boldsymbol{\alpha}_{XOR(u_4, v_4)}) = 2$.

The first vector shows that both of u_2 and v_2 write on D_1 and read D_2. In this vector, the corresponding element to read from D_1 is 0, because D_1 is read only by activity v_2. The second vector says that D_1 and D_3 are written and D_2 is read by u_2 or v_4. Finally, the third vector shows that D_3 is written by only on u_4 or v_4. In this vector, the corresponding element to write on D_1 is 0, because D_1 is written by both activities.

3 Measuring similarity between activities

In this section, a metric to measure the similarity of activities is proposed. This metric is based on data and behaviour similarities of activity nodes. Where data similarity is defined based on the activities' data accesses, behaviour similarity is related to the position of those activities inside processes.

Two activities are similar from the data point of view, if they have access to similar sets of data objects. To measure the data similarity of two activities, we use activity vectors. As defined in the previous section, *AND* operation can be used to measure the share access of two activities. In addition, we need to compute *XOR* of those activities to measure the differences between them. So, basically data similarity of two activities is defined as the weight of shared access divided by the weight of all data accesses of two activities.

Definition: (Data Similarity): Let $G = (N, s, F, V, D, O)$ and $G' = (N', s', F', V', D', O')$ be two graphs and $t \in (N \cap U_A)$, $t' \in (N' \cap U_A)$ be two activities such that $\boldsymbol{\alpha}_t = (u_1, u_2, \ldots, u_{2m})$ and $\boldsymbol{\alpha}_{t'} = (v_1, v_2, \ldots, v_{2m})$.

The value of *data similarity* between two activities is defined as follows:

$$Sim^d(t, t') = \frac{2 \times w(\boldsymbol{\alpha}_{AND(t,t')})}{2 \times w(\boldsymbol{\alpha}_{AND(t,t')}) + w(\boldsymbol{\alpha}_{XOR(t,t')})} \quad (1)$$

Example 5: Continuing with Example 2, following are data similarities that are calculated on some activity pairs:

- $Sim^d(u_1, v_1) = \dfrac{2 * w(\boldsymbol{\alpha}_{AND(u_1, v_1)})}{2 * w(\boldsymbol{\alpha}_{AND(u_1, v_1)}) + w(\boldsymbol{\alpha}_{XOR(u_1, v_1)})} = \dfrac{2 \times 4}{2 \times 4 + 0} = 1,$
- $Sim^d(u_5, v_3) = \dfrac{2 \times 0}{2 \times 0 + 4} = 0,$
- $Sim^d(u_3, v_5) = \dfrac{2 \times 2}{2 \times 2 + 2} = 0.66.$

Table 1 shows data similarity between activities of *Loan* processes of Figs. 1 and 2.

The next similarity measure is behavioural similarity. Behavioural similarity between two activities is defined based on the structure of processes. To find similar activities within different business processes, the positions of those activities are taken into account. Each activity within a process has a set of successor and a set of predecessor activities. Also it can have a set of tasks that may be executed in parallel with. The similar situation exists for conditional positions. Moreover, finally tasks can be executed in loop situations.

Definition: (Structural Relations between Activities): Let $G = (N, s, F, V, D, O)$ be a Business Process Graph, for each activity node $t \in (N \cap U_A)$:

- $\sigma(t)$ is a set of successor activity node, associating to activity node t, a set of activity nodes in G.
- The transitive closure of the successor function, denoted as σ^*

$$\sigma^*(t) = \{p \mid p \in \sigma(t) \vee (\exists s : s \in \sigma(t) \wedge p \in \sigma^*(s))\}$$

- $\pi(t)$ is a set of predecessor activity nodes represents the set of activity nodes in G that have t as successor

$$\pi(t) = \{s \mid t \in \sigma(s)\}$$

- The transitive closure of the predecessor function, denoted as π^*

$$\pi^*(t) = \{p \mid p \in \pi(t) \vee (\exists s : s \in \pi(t) \wedge p \in \pi^*(s))\}$$

- $\chi(t)$ is a set of activities that are mutual exclusive with t,
- $\gamma(t)$ is a set of activities that can be executed in parallel with t,
- $\phi(t)$ is a set of activities that are in a same loop with t.

Example 6: Continuing with Example 2, the following sets show the different dependency sets of some activities:

- $\sigma(u_2) = \{u_3, u_4\}$, $\sigma^*(u_2) = \{u_3, u_4, u_5\}$, $\pi(u_2) = \{u_1\}$, $\pi^*(u_2) = \{u_1\}$, $\chi(u_2) = \varnothing$, $\gamma(u_2) = \varnothing$, $\phi(u_2) = \varnothing$,
- $\sigma(u_3) = \{u_5\}$, $\sigma^*(u_2) = \{u_5\}$, $\pi(u_3) = \{u_2\}$, $\pi^*(u_3) = \{u_1, u_2\}$, $\chi(u_3) = \{u_4\}$, $\gamma(u_2) = \varnothing$, $\phi(u_2) = \varnothing$,
- $\sigma(v_1) = \{v_2, v_5\}$, $\sigma^*(v_1) = \{v_2, v_3, v_4, v_5, v_6\}$, $\pi(v_1) = \varnothing$, $\pi^*(v_1) = \varnothing$, $\chi(v_1) = \varnothing$, $\gamma(v_1) = \varnothing$, $\phi(v_1) = \varnothing$,
- $\sigma(v_3) = \{v_4, v_5\}$, $\sigma^*(v_3) = \{v_4, v_5, v_6\}$, $\pi(v_3) = \{v_2\}$, $\pi^*(v_3) = \{v_2, v_1\}$, $\chi(v_3) = \varnothing$, $\gamma(v_3) = \varnothing$, $\phi(v_3) = \varnothing$.

Based on these different sets that are related to the structure of a process, behavioural similarity can be defined. In fact, behavioural similarity considers both the behaviour and the structure of a process. To this end, all the previous sets are considered and based on the differences between these sets for any pair of activities; the behavioural similarity for that pair is calculated.

Definition: (Behavioural Similarity): Let $G = (N, s, F, V, D, O)$ and $G' = (N', s', F', V', D', O')$ be two graphs and $t \in (N \cap U_A)$, $t' \in (N' \cap U_A)$ be two activity node such that $\boldsymbol{\alpha}_t = (u_1, u_2, \ldots, u_{2m})$, $\boldsymbol{\alpha}_{t'} = (v_1, v_2, \ldots, v_{2m})$. The value of *behavioural similarity* between activity nodes t and t' is defined based on the sum of the differences of corresponding sets (see (2))

Example 7: Continuing with Example 2, the behavioural similarity of some activities is shown below:

Table 1 Data similarity between activities of the loan processes

	v_1	v_2	v_3	v_4	v_5	v_6
u_1	1	1	0.57	0.5	0.66	0.4
u_2	0.86	0.86	0.66	0.66	0.8	0
u_3	0.5	0.5	0.57	1	0.66	0
u_4	0.66	0.66	0.4	0.66	1	0
u_5	0.4	0.4	0	0	0	1

- $sim^b(u_2, v_1) = 1 - ((|1-0| + |3-5| + 0 + 0 + 0)/(5+6))$, $= 0.73$
- $sim^b(u_3, v_3) = 1 - ((|2-2| + |1-3| + |1-0| + 0 + 0)/(5+6)) = 0.73$.

Table 2 shows behavioural similarity between activities of *Loan* processes of Figs. 1 and 2.

Finally, the average of data similarity and behavioural similarity is considered as the integrated similarity.

Definition: (Integrated Similarity): Let $G = (N, s, F, V, D, O)$ and $G' = (N', s', F', V', D', O')$ be two graphs and $t \in (N \cap U_A)$, $t' \in (N' \cap U_A)$ be two activity nodes. Integrated similarity of t and t' can be defined as

$$integrated\_Sim(t, t') = \frac{Sim^d(t, t') + Sim^b(t, t')}{2} \tag{3}$$

Example 8: Table 3 shows the integrated similarity between activities of the loan processes.

4 Measuring similarity between processes

In this section, the proposed algorithm to calculate the similarity of the two processes is introduced. This algorithm calculates the similarity of two processes using the similarity of activity pairs.

To begin with, let define an enumeration for process activities. Let G be a graph with $size(G) = n$, the sequence of activities $a_1, a_2, \ldots, a_n$ is a valid(fixed) *enumeration* for activity nodes within G, when for all i, j, if $i \leq j$ and $i \notin \phi(j)$ then $a_j \notin \pi^* a_i$.

The algorithm takes two processes (one as the base) and enumeration of activities in each process as input and outputs a number as similarity of those two processes.

First, three sets called *New*, *Open*, and *Closed* are defined. The *new* set initially contains *start*, which is the start event of the base process and *Open* and *Closed* are empty. *Start* node is taken out of the *New* set and two matchings with the largest *integrated_sim* of the first activity in the activity enumeration of the base process [e.g. (v_1, v_m), (v_1, v_n)] are added to *New*, also *start* is added to the *Open* set. *Sim_score* of these two pairs is calculated based on their *integrated_sim*. Then the element with the largest *sim_score* value [e.g. (v_1, v_m)] is removed from *New* and is added to *Open* set and instead of that element in the two matchings with the largest values of the next activity (v_2), are added to the tree as the children of v_1. They are also added to *New* set [e.g. (v_2, v_p), (v_2, v_q)]. For these two elements, *sim_score* is calculated based on their *integrated_sim* ant their parent's *sim_score*. At this time, the *New* set contains three pairs: (v_2, v_p), (v_2, v_q), and (v_1, v_n) and the *Open* set contains *start* and (v_1, v_m). Once again the largest element of *New* set is chosen and added to *Open*. If the chosen element of *New* set is the last existing child, no other siblings exist in *New* and the parent node is in *Open* set (is not completely extended) the next largest sibling is added to *New*. For example, if (v_1, v_n) is the largest element of *New*, after removing it from *New*, since it does not have any other siblings such as (v_1, v_x) and the *start* is still in *Open*, the next child of *start* with the largest value for $sim\_score(v_1, v_k)$ is added to *New*.

For each element in *Open* set, a counter is defined that counts the extended children. When all children are extended, the node is removed from *Open* set and is added to *Closed* set. The algorithm continues until when the largest element of the *New* set is a matching of the last activity of the base process or in other words no extension is possible. In this case, the *sim_score* is returned as the similarity of two processes. Algorithm 1 (see Fig. 5) shows the proposed algorithm.

Although the worst case of the proposed algorithm has the same complexity as the worst case of A^* algorithm, for other cases the complexity is different. For example, for the best case, the complexity of our algorithm in equal to $2n$, while the A^* needs to calculate $(n(n+1)/2)$ nodes.

Example 9: Fig. 6 shows the results of performing Algorithm 1 (Fig. 5) on the loan business processes graphs (Figs. 3 and 4). The grey nodes show the best matching. Let's consider the first loan process (Fig. 3) as the base one. Node u_1 is the first node to extend. The algorithm adds the two largest matchings of this activity, which are (u_1, v_1) and (u_1, v_2), and their corresponding similarity scores to the tree. Then the first one, here (u_1, v_1), is extended and two largest matchings of the next activity are added to the tree and their similarity scores are calculated. In this example, (v_2, u_2) and (v_2, v_3) are the best matchings. Now, except root, the tree has four nodes; (u_1, v_1) and (u_1, v_2) as children of the root node and (u_2, v_2) and (u_2, v_3) as children of the (u_1, v_1) node where (u_1, v_1) is in the Open set and other three nodes are in the New set. Next, the algorithm chooses the node in the New set with the largest similarity score, which is (u_1, v_2), extends it by adding its two most appropriate children to the tree and moves it to the Open set. Here (u_2, v_1) and (v_2, v_3) both with score 0.948 are added to the New node. Again the node in the New set with the largest similarity score ((u_2, v_2), 0.968) is selected and extended. The algorithm continues until node (u_5, v_6) is added as the child of (u_4, v_5) and since it is the

$$Sim^b(t, t') = 1 - \frac{||\sigma^*(t)| - |\sigma^*(t')|| + ||\pi^*(t)| - |\pi^*(t')|| + ||\chi(t)| - |\chi(t')|| + ||\gamma(t)| - |\gamma(t')|| + ||\phi(t)| - |\phi(t')||}{|G| + |G'|} \tag{2}$$

Table 2 Behaviour similarity between activities of the loan processes

	v_1	v_2	v_3	v_4	v_5	v_6
u_1	0.91	0.91	0.73	0.36	0.36	0.18
u_2	0.73	0.91	0.91	0.55	0.55	0.36
u_3	0.36	0.55	0.73	0.91	0.91	0.55
u_4	0.36	0.55	0.73	0.91	0.91	0.55
u_5	0.18	0.36	0.55	0.73	0.73	0.91

Table 3 Integrated similarity between activities of the loan processes

	v_1	v_2	v_3	v_4	v_5	v_6
u_1	0.95	0.95	0.65	0.43	0.52	0.29
u_2	0.79	0.89	0.79	0.61	0.67	0.18
u_3	0.43	0.52	0.65	0.95	0.79	0.27
u_4	0.52	0.61	0.76	0.79	0.95	0.27
u_5	0.29	0.38	0.27	0.36	0.36	0.95

Input: $G = (N, s, F, V, D, O)$ and $G' = (N', s', F', V', D', O')$ s.t $(u_1, u_2, ..., u_{|G|})$ and $(v_1, v_2, ..., v_{|G'|})$ be the fixed enumerations of activities in G and G' respectively
Output: Similarity of two business processes

New := {start} ▷ The set of nodes to be evaluated, initially containing the start node
Open := {} ▷ The set of nodes that are not completely evaluated.
Closed := {} ▷ The set of nodes already evaluated.
Came_From := the empty path ▷ The path of navigated nodes.
$Open.Insert(start)$
$New.Remove(start)$
$Count_{start} = 0$
Let (u_1, v_p) and (u_1, v_q) be two matchings of the activity u_1 with the largest $integrated\_sim$
$New.Insert(u_1, v_p)$
$New.Insert(u_1, v_q)$
$sim\_score(u_1, v_p) = 1 - \frac{1-integreted\_sim(u_1,v_p)}{|G|}$
$sim\_score(u_1, v_q) = 1 - \frac{1-integreted\_sim(u_1,v_q)}{|G|}$
$i = 1$
while $Open$ is not empty **do**
current := the node (u_i, v_j) in New having the maximum sim_score value
if $i = size(G)$ **then**
return $reconstruct\_path(Came\_From, current)$
end if
$Open.Insert(current)$
$New.Remove(current)$
$Count_{current} = i$
Count$_{current's\ parent}$ = *Count*$_{current's\ parent}$ *+1*
if *Count*$_{current's\ parent}$ = $|G'|$ **then**
Close.Insert(current's parrent)
Open.Remove(current's parrent)
end if
if there is no other *current's sibling* in New and the *current's parent* is still in $open$ set **then**
Let node (u_i, v_k) be the *current's sibling* with the largest *integrated-sim* value and $(u_i, v_k) \notin (New \cup Open \cup Closed)$
$New.Insert(u_i, v_k)$
$sim\_score[(u_i, v_k)] = sim\_score$*(current's parent)* $- \frac{1-integrated\_sim(u_i,v_k)}{|G|}$
end if
Let (u_{i+1}, v_r) and (u_{i+1}, v_s) be two matching of the *current's children* with the large $integrated\_sim$ such that v_j is not in $Come\_From^*[current]$
$New.Insert(u_{i+1}, v_r)$
$New.Insert(u_{i+1}, v_s)$
Came_From[(u_{i+1}, v_r)] = Came_From[(u_{i+1}, v_s)] := $current$
$sim\_score[(u_{i+1}, v_r)] := sim\_score(current) - \frac{1-integrated\_sim(u_{i+1},v_r)}{|G|}$
$sim\_score[(u_{i+1}, v_s)] := sim\_score(current) - \frac{1-integrated\_sim(u_{i+1},v_s)}{|G|}$
i = i+ 1
end while
return failure
function RECONSTRUCT_PATH(Came_From,current)
total_path := [current]
while current in Came_From.Keys: **do**
current := Came_From[current]
total_path.append(current)
end while
return total_path
end function

Fig. 5 *Algorithm 1: Measuring similarity of two business processes*

matching for the last node of the base process (u_5) and there is no node in New set with the larger similarity score, the algorithm is finished and returns the similarity score of the last node as the output.

As it can be seen the similarity of those two processes is equal to **0.938**.

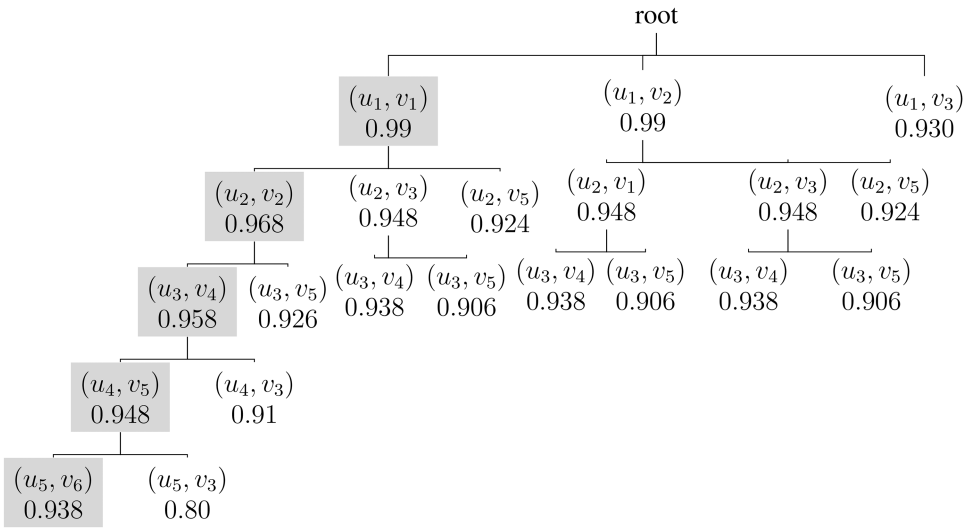

Fig. 6 *Measuring similarity between the loan processes*

5 Discussion and evaluation

In this part, the proposed method is evaluated using two different methods. First, ten different real processes from four different businesses are considered as base processes and for each of those processes, four other variants are taken into account. Variants are process models which are similar to the base process and are modelled to achieve the same business goal. Most of these variants are obtained from papers, reference models, and real businesses and some others are designed manually. By finishing this step, basically we had 50 models from 10 different processes where different variants of a process have access to same data model. Since the origin of these processes is different, we had to manually modify their data model. Basically, we modify the name of data objects to be unified. Then, using the proposed method the similarity of each base process model and its four variants are measured. Finally, we asked experts of those businesses to rank these variants based on the similarity with the base model. For each process three experts are asked to rank variants separately. In case their rankings were different, we ask them to discuss and agree on rankings. Form these ten processes, in two cases, experts did not agree even after negotiation, so we just consider the average of all three rankings and rank the variants. Table 4 shows the resulted similarity scores. Each row belongs to one process and its variants. For each variant we specify the similarity score and then two different ranks. The first rank is based on the similarity score and the second rank is the result of experts ranking. For example, for process P_2, similarity scores for variants 1, 2, 3, and 4, are 0.88, 0.84, 0.82, and 0.45%, respectively. Since the variant 1 has the maximum score, its rank is 1. Similarly, variants 2, 3, and 4 ranks are 2, 3, and 4. However, from the experts point of view variant 3 has the most similarity to the base model, the rank is 1, variant 1 is rank 2, variant 2 is rank 3, and finally variant 4 is rank 4.

Next, we want to quantify the conformance of our method with experts' opinions using the resulting ranks from both the proposed method and experts. To do this, if the variant with the maximum similarity score is identified correctly, the method gains four points. This point for ranks 2, 3, and 4 variants are 3, 2, and 1, respectively. Also, if we just need to substitute ranks 1 and 2 variants, the method gains four points (from seven total points), for the substitution of ranks 2 and 3, the method gains three points (instead of five total points) and it gains two points if we need to substitute ranks 3 and 4 variants.

For instance, in the above example, since there are four variants for each process, the maximum gained points can be 10 per process. So, process P_1 gets nine points, since it ranks variants 1 and 2 correctly and we just need to substitute variants 3 and 4. However, process P_2 gains only one point, since the first, second, and third variants are ranked incorrectly. The last column of Table 4 represents the score of each process where the proposed method gets totally *82 scores out of 100*. The promising aspect of the results is that the faults occur only when the similarity scores of variants are very close to each other, e.g. in process P_2, the difference between the similarity score of the first and the second instances is just 0.03.

The next evaluation is done to figure out the precision, recall and accuracy of the proposed method. We again consider the above-mentioned processes. In this time, we defined a threshold for similar processes and find the variant with similarity more than it. To define this threshold based on experts' ideas and results from different businesses, we selected 80% as an acceptable threshold for process similarity. Then we asked experts to find the variants of each process (from those four variants) with similarity more than the defined threshold, and based on these results we figured out the precision and recall of our method. Table 5 shows the result of our method and experts' opinions. In this table, *V* stands for *Variant*. For example, for process P_2, our method detects variants 1, 2, and 3 as the set of variants which are >80% similar to the base model, whereas the experts only detects variants 1 and 2. There are totally

Table 4 Measuring the similarity of different processes

Base process	Variant 1		Variant 2		Variant 3		Variant 4		Score(10)
process P_1	0.93	1 1	0.84	2 2	0.65	3 4	0.61	4 3	9
process P_2	0.88	1 2	0.84	2 3	0.82	3 1	0.45	4 4	1
process P_3	1	1 1	0.81	2 2	0.68	3 3	0.62	4 4	10
process P_4	0.94	1 2	0.91	2 1	0.72	3 3	0.64	4 4	7
process P_5	0.73	1 1	0.54	2 2	0.48	3 4	0.44	4 3	9
process P_6	0.91	1 1	0.84	2 2	0.70	3 3	0.54	4 4	10
process P_7	0.79	1 1	0.75	2 3	0.72	3 2	0.57	4 4	8
process P_8	0.96	1 1	0.93	2 2	0.87	3 4	0.87	4 3	9
process P_9	0.98	1 1	0.91	2 2	0.66	3 4	0.63	4 3	9
process P_{10}	0.66	1 1	0.58	2 2	0.47	3 3	0.44	4 4	10

40 variants for 10 processes that can be categorised into four groups based on the results from the proposed method:

i. Similar variants that are detected as similar by the proposed method (true positive – TP),
ii. Similar variants that are detected as non-similar by the proposed method (false negative – FN),
iii. Non-similar variants that are detected as similar by the proposed method (false positive – FP), and
iv. Non-similar variants that are detected as non-similar by the proposed method (true negative – TN).

Table 6 shows these four metrics which help us define precision, recall, and accuracy, respectively,

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{13}{13 + 4} = 0.76 \tag{4}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{13}{13 + 2} = 0.87 \tag{5}$$

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{13 + 21}{13 + 21 + 4 + 2} = 0.85 \tag{6}$$

6 Related work

In this section, we discuss related work in two areas. First, methods to map two different activities are presented, then we focus on measuring the similarity between models that can be computed using the similarity of activities. These two areas are two main steps of comparing two business processes [21].

6.1 Mapping between two process activities

To calculate the similarity of two process models or rank processes that satisfy a given query, first the similarity of activities must be considered as the basis. Different parameters and methods can be applied to measure elements similarity. Some approaches intend to map activities based on experts' ideas [22]. Others try to use a combination of both methods; first, they use an automatic method to map activities, then final results are evaluated by experts [23]. Three ways of measuring elements similarity are mentioned in [3] which are mainly based on the labels: syntactic similarity, semantic similarity, and contextual similarity. Beside activity label, the role of activity is used to calculate the similarity of nodes in [24, 25]. In those paper role refers to the neighbour nodes in process and can be of forms start, stop, split, join, and regular roles. Similar to many approaches, Dijkman *et al.* [26] used string edit distance to compute node similarity. Baumann *et al.* [11] involved actors, data objects, and their order of appearing in the model to match activities based on their labels. The proposed mapping is partial and injective. Also Klinkmüller *et al.* [27] used label matching to find similarity activities. The method works based on the bag-of-word technique. An algorithm for determining correspondences between activities using PST is introduced in [28]. The algorithm has two phases, first establishing correspondences based on similarity of model element attributes such as types and names and then refining the result based on the structure of the models. Also Montani *et al.* [29] used a semantic approach to compare process activities by making use of domain knowledge. It takes into account complex control flow constructs (such as AND and XOR splits/joins). Leopold *et al.* [30] used another approach based on semantic techniques and probabilistic optimisation The ICoP framework [31] focuses on matchers that also detect complex correspondences between groups of activities, where existing matchers focus on 1:1 correspondences.

While most of the proposed methods are based on labels, label-based matching probably fails, e.g. when modellers use different vocabulary. So it is useless or even impossible to use only label matching to analyse sets of activities [11].

Table 5 Finding similar processes' variants

Base process	Similar variants by method	Similar variants by expert
process 1	{V1, V2}	{V1}
process 2	{V1, V2, V3}	{V1, V2}
process 3	{V1, V2}	{V1}
process 4	{V1, V2}	{V1, V2}
process 5	{ }	{V1}
process 6	{V1, V2}	{V1}
process 7	{ }	{V1}
process 8	{V1, V2, V3, V4}	{V1, V2, V3, V4}
process 9	{V1, V2}	{V1, V2}
process 10	{V1}	{V1}

Table 6 Measuring precision and recall in sample processes

Metrics	Number of variants
similar variants detected by method as similar (TP)	13
similar variants detected by method as non-similar (FN)	2
non-similar variants detected by method as similar (FP)	4
non-similar variants detected by method as non-similar (TN)	21

6.2 Calculating the similarity with maximal mapping

After mapping activities either manually or automatically, it is time to calculate the similarity of processes. The similarity measuring procedure can be performed mainly in two forms:

i. The similarity of the two processes is computed. This can be the computation of similarity between process pairs in a same repository to find similar ones or the computation of similarity between a given process with processes in a repository and then ranking processes based on their similarity with the given process. Actually, there are different similarity measures that are analysed in [32].
ii. Some specifications are expressed in a form of query, then processes which hold those specifications are ranked based on their similarity with the given query [33]. Whatever the form of similarity measuring is, common approaches with minor changes can be applied for different objectives.

To calculate the similarity of processes, methods mostly use either structural or behavioural similarity. Structural methods consider the business process as a graph in which nodes of graph are tasks and use techniques such as graph edit distance (GED) [34] and maximal common sub-graph (MCS) [35] to find the similarity. The GED measures the distance between graphs by using a set of editing operations. The MCS measures the distance by examining the difference between minimal common super-graph and maximal common sub-graph. Similarity methods using behavioural measure consider the order of execution of tasks as behaviour. A similar idea as GED is used in [24] by considering the structural features for each process and compute the number of features in one process model that are matched by features of another process. In [3, 26] text similarity based on pairwise comparisons of node labels, structural similarity using graph edit similarity and behavioural similarity by considering indirect relations in structural similarity are introduced to find similar processes. There are several researches in domain of behavioural similarity. A behavioural process similarity algorithm named CFS based on complete firing sequences is proposed in [2]. Complete firing sequences are used to express model behaviour. To find an optimal mapping between two sets of complete firing sequences A^*

algorithm is used. Given two process models, Armas-Cervantes *et al.* [36] determined if they are behaviourally equivalent, and if not, it describes their differences in terms of behavioural relations. The technique is based on a translation from process models to event structures, a formalism that describes the behaviour as a collection of events (task instances) connected by binary behavioural relations. Kunze *et al.* [37] present a proper metric to quantify process similarity based on behavioural profiles grounded in the Jaccard coefficient, which leverages behavioural relations between pairs of process model activities. A method to compare a given query with candidate process models using few relevant aspects expressed by a user is presented in [38]. It finds the ones that share common features with the query. The precedence relation and the weak order relation are introduced to model behavioural relations. Approximate clone detection introduced in [39] is the process of identifying similar process fragments in business process model collections. The tool presented in this paper cluster approximate clones in large process model repositories. Another clustering method to compute process models similarity is introduced in [40].

Since most approaches for process similarity are either structural or behavioural, they suffer from exponential complexity, so the abstractions are proposed in [33] in which the order of execution of two activities is considered. Indexing techniques can be applied to implement searching for all similar processes to a query process, more efficient. The indexing can be done in two ways: indexing process elements and indexing complete process models. To reduce the comparisons required for finding desired models in a repository an indexing approach based on metric trees, a hierarchical search structure is proposed [41].

7 Discussion and conclusion

This paper is focused on business process similarity and proposed a method for calculating the similarity of processes. The proposed method calculates the similarity in two steps: First between activity pairs and then between process pairs. Existing methods use activity labels to find the similar activities, while these labels are ambiguous, inexact and even meaningless. To overcome this problem and due to the use of data in process modelling in the recent decades, the proposed method uses data features along with structural and behavioural features to calculate the similarity of two processes. To this end, the read and write accesses of activities to same data objects are considered as the basis of data similarity. Also the position of an activity within a process and its connections with other nodes are taken into account to obtain a more accurate similarity metric. In the next step, the proposed algorithm is used to calculate the similarity of two processes. This algorithm finds the best matching between activities while trying to minimise the number of visited activities. Our evaluations showed a satisfactory result in comparison with other methods especially when data accesses are specified and activity labels are not appropriate.

7.1 Limitations of the method

There are some limitations and restrictions for the proposed method. First, we do not support every business process models. The method basically supports BPMN processes where no event is defined in the model.

Second, we measure the similarity of processes pair, means if we want to measure the similarity of every processes pair in a data set; we need to run the algorithm for each pair separately.

The third limitation is about data. From the data point of view, we model object read and write while in today's workflow systems data is not only a set of objects which are read or written. Each data object has several attributes, can be in different states and affects a business process a lot. Therefore, we need to have a deep look at role of data in business processes. In addition, when data objects are taken into account measuring the similarity between different objects which are accessible from processes could be an interesting problem.

7.2 Future work

As future work, activities with different granularities can be taken into account. These activities can be identified by considering their data accesses [11]. Also we are working on calculating the similarity of different data objects. To this end attribute labels and types, also the relationship between objects (e.g. foreign keys) can be used [42–45]. Adding these features to the proposed method can provide a complete framework for calculating the similarity of processes.

8 References

[1] Documentair structuurplan, http://www.model-dsp.nl/, last accessed: 20 September 2016

[2] Dong, Z., Wen, L., Huang, H., *et al.*: 'CFS: a behavioral similarity algorithm for process models based on complete firing sequences', in '*OTM: on the move to meaningful internet systems*' (Springer Berlin Heidelberg, 2014) (LNCS, **8841**), pp. 202–219

[3] Dijkman, R., Dumas, M., Van Dongen, B., *et al.*: 'Similarity of business process models: metrics and evaluation', *Inf. Syst.*, 2011, **36**, (2), pp. 498–516

[4] Van Dongen, B., Dijkman, R., Mendling, J.: 'Measuring similarity between business process models', in '*Advanced information systems engineering*' (Springer Berlin Heidelberg, 2008) (LNCS, **5074**), pp. 450–464

[5] Rahm, E., Bernstein, P.A.: 'A survey of approaches to automatic schema matching', *VLDB J.*, 2001, **10**, (4), pp. 334–350

[6] Euzenat, J., Shvaiko, P.: '*Ontology matching*' (Springer-Verlag, Heidelberg (DE), 2007), vol. **18**

[7] Levenshtein, V.I.: 'Binary codes capable of correcting deletions, insertions, and reversals', *Soviet Phys. Doklady*, 1966, **10**, (8), pp. 707–710

[8] Salton, G., Wong, A., Yang, C.S.: 'A vector space model for automatic indexing', *Commun. ACM*, 1975, **18**, (11), pp. 613–620

[9] Manning, C.D., Schütze, H.: '*Foundations of statistical natural language processing*' (MIT press, Cambridge, 1999), vol. **999**

[10] Miller, G.A.: 'WordNet: a lexical database for English', *Commun. ACM*, 1995, **38**, (11), pp. 39–41

[11] Baumann, M.H., Baumann, M., Schönig, S., *et al.*: 'Towards multi-perspective process model similarity matching', in '*Enterprise and organizational modeling and simulation*' (Springer Berlin Heidelberg, 2014) (LNBIP, **191**), pp. 21–37

[12] Kunzle, V., Reichert, M.: 'PHILharmonicFlows: towards a framework for object-aware process management', *J. Softw. Maint. Evol., Res. Pract.*, 2011, **23**, (4), pp. 205–244

[13] Meyer, A., Pufahl, L., Fahland, D., *et al.*: 'Modeling and enacting complex data dependencies in business processes', in '*Business process management*' (Springer Berlin Heidelberg, 2013) (LNCS, **8094**), pp. 171–186

[14] Hull, R., Su, J., Vaculín, R.: 'Data management perspectives on business process management: tutorial overview'. Proc. of the 2013 ACM SIGMOD Int. Conf. on Management of Data, June 2013, pp. 943–948

[15] Reichert, M.: 'Process and data: two sides of the same coin', in '*20th Int'l Conf on Cooperative Information Systems (CoopIS'12), OTM 2012, Part I, September 12–14, 2012, Rome, Italy*' (DBIS EPub)

[16] Weske, M.: '*Business process management: concepts, languages, architectures*' (Springer Science & Business Media, 2012)

[17] Amiri, M.J., Parsa, S., Mohammad Zade Lajevardi, A.: 'Multifaceted service identification: process, requirement and data', *Comput. Sci. Inf. Syst.*, 2016, **13**, (2), pp. 335–358

[18] Ebrahimifard, A., Amiri, M.J., Arani, M.K., *et al.*: 'Mapping BPMN 2.0 choreography to WS-CDL: a systematic method', *J. E-Technol.*, 2016, **7**, (1), pp. 1–23

[19] Amiri, M.J., Koupaee, M.: 'Quality improvement in web services using function replication'. Second Int. Conf. on Web Research (ICWR), Tehran, Iran, April 2016, pp. 72–77

[20] Weber, I., Hormann, J., Mendling, J.: 'Semantic business process validation'. SBPM'08: Third Int. Workshop on Semantic Business Process Management at ESWC'08, 2008

[21] Becker, M., Laue, R.: 'A comparative survey of business process similarity measures', *Comput. Ind.*, 2012, **63**, (2), pp. 148–167

[22] Dijkman, R.: 'Diagnosing differences between business process models', in '*Business process management*' (Springer Berlin Heidelberg, 2008) (LNCS, **5240**), pp. 261–277

[23] Ehrig, M., Koschmider, A., Oberweis, A.: 'Measuring similarity between semantic business process models'. Proc. of the Fourth Asia-Pacific Conf. on Conceptual Modeling, 2007, vol. **67**, pp. 71–80

[24] Yan, Z., Dijkman, R., Grefen, P.: 'Fast business process similarity search with feature-based similarity estimation', in '*OTM: on the move to meaningful internet systems*' (Springer Berlin Heidelberg, 2010) (LNCS, **6426**), pp. 60–77

[25] Yan, Z., Dijkman, R., Grefen, P.: 'Fast business process similarity search', *Distrib. Parallel Databases*, 2012, **30**, (2), pp. 105–144

[26] Dijkman, R., Dumas, M., García-Bañuelos, L.: 'Graph matching algorithms for business process model similarity search', in '*Business process management*' (Springer Berlin Heidelberg, 2009) (LNCS, **5701**), pp. 48–63

[27] Klinkmüller, C., Weber, I., Mendling, J., *et al.*: 'Increasing recall of process model matching by improved activity label matching', in '*Business process management*' (Springer Berlin Heidelberg, 2013) (LNCS, **8094**), pp. 211–218

[28] Branco, M.C., Troya, J., Czarnecki, K., *et al.*: 'Matching business process workflows across abstraction levels'. Int. Conf. on Model Driven Engineering Languages and Systems, September 2012, pp. 626–641

[29] Montani, S., Leonardi, G., Quaglini, S., *et al.*: 'A knowledge-intensive approach to process similarity calculation', *Expert Syst. Appl.*, 2015, **42**, (9), pp. 4207–4215
[30] Leopold, H., Niepert, M., Weidlich, M., *et al.*: 'Probabilistic optimization of semantic process model matching', in '*Business process management*' (Springer Berlin Heidelberg, 2012) (LNCS, **7481**), pp. 319–334
[31] Weidlich, M., Dijkman, R., Mendling, J.: 'The ICoP framework: identification of correspondences between process models', in '*Advanced information systems engineering*' (Springer Berlin Heidelberg, 2010) (LNCS, **6051**), pp. 483–498
[32] Becker, M., Laue, R.: 'Analysing differences between business process similarity measures'. Business Process Management Workshops, 2012, pp. 39–49
[33] Dijkman, R., Van Dongen, B., Dumas, M., *et al.*: 'A short survey on process model similarity', in '*Seminal contributions to information systems engineering*' (Springer Berlin Heidelberg, 2013), pp. 421–427
[34] Zhang, K., Shasha, D.: 'Simple fast algorithms for the editing distance between trees and related problems', *SIAM J. Comput.*, 1989, **18**, (6), pp. 1245–1262
[35] Bunke, H., Shearer, K.: 'A graph distance metric based on the maximal common subgraph', *Pattern Recognit. Lett.*, 1989, **19**, pp. 255–259
[36] Armas-Cervantes, A., Baldan, P., Dumas, M., *et al.*: 'Diagnosing behavioral differences between business process models: an approach based on event structures', *Inf. Syst.*, 2016, **56**, pp. 304–325
[37] Kunze, M., Weidlich, M., Weske, M.: 'Behavioral similarity – a proper metric', in '*Business process management*' (Springer Berlin Heidelberg, 2011) (LNCS, **6896**), pp. 166–181
[38] Kunze, M., Weske, M.: 'Local behavior similarity', in '*Enterprise, business-process and information systems modeling*' (Springer Berlin Heidelberg, 2012) (LNBIP, **113**), pp. 107–120
[39] La Rosa, M., Dumas, M., Ekanayake, C., *et al.*: 'Detecting approximate clones in business process model repositories', *Inf. Syst.*, 2015, **49**, pp. 102–125
[40] Niemann, M., Siebenhaar, M., Schulte, S., *et al.*: 'Comparison and retrieval of process models using related cluster pairs', *Comput. Ind.*, 2012, **63**, (2), pp. 168–180
[41] Kunze, M., Weske, M.: 'Metric trees for efficient similarity search in large process model repositories', in '*Business process management workshops*' (Springer Berlin Heidelberg, 2011) (LNBIP, **66**), pp. 535–546
[42] López, M.T.G., Borrego, D., Gasca, R.M.: 'Data state description for the migration to activity-centric business process model maintaining legacy databases', in '*International Conference on Business Information Systems*' (Springer, Cham, 2014) (LNBIP, **176**), pp. 86–97
[43] Meyer, A., Pufahl, L., Batoulis, K., *et al.*: 'Automating data exchange in process choreographies', *Inf. Syst.*, 2015, **53**, pp. 296–329
[44] Meyer, A., Pufahl, L., Fahland, D., *et al.*: 'Modeling and enacting complex data dependencies in business processes', in '*Business Process Management*' (Springer, Berlin, Heidelberg, 2013) (LNCS, **8094**), pp. 171–186
[45] Gómez-López, M.T., Perez-Álvarez, J.M., Varela-Vaca, Á.J., *et al.*: 'Guiding the creation of choreographed processes with multiple instances based on data models'. Business Process Management Workshops – BPM 2016, 14th Int. Workshops, Rio, Brazil, 18–22 September 2016