Information Systems 36 (2011) 498–516

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Similarity of business process models: Metrics and evaluation

Remco Dijkman [a,*], Marlon Dumas [b], Boudewijn van Dongen [c], Reina Käärik [b], Jan Mendling [d]

[a] School of Industrial Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[b] Institute of Computer Science, University of Tartu, J Liivi 2, 50409 Tartu, Estonia
[c] Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
[d] Institute of Information Systems, Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany

ARTICLE INFO

Article history:
Received 15 April 2009
Received in revised form
9 September 2010
Accepted 20 September 2010
Recommended by: M. Weske

Keywords:
Business process management
Process similarity
Process model repository
Process model search

ABSTRACT

It is common for large organizations to maintain repositories of business process models in order to document and to continuously improve their operations. Given such a repository, this paper deals with the problem of retrieving those models in the repository that most closely resemble a given process model or fragment thereof. Up to now, there is a notable research gap on comparing different approaches to this problem and on evaluating them in the same setting. Therefore, this paper presents three similarity metrics that can be used to answer queries on process repositories: (i) node matching similarity that compares the labels and attributes attached to process model elements; (ii) structural similarity that compares element labels as well as the topology of process models; and (iii) behavioral similarity that compares element labels as well as causal relations captured in the process model. These metrics are experimentally evaluated in terms of precision and recall. The results show that all three metrics yield comparable results, with structural similarity slightly outperforming the other two metrics. Also, all three metrics outperform text-based search engines when it comes to searching through a repository for similar business process models.

1. Introduction

Many organizations have built over time repositories of business process models that serve as a knowledge base for their ongoing business process management efforts. Such repositories may contain hundreds or even thousands of business process models. For example, we have access to a repository of the Dutch local governments council containing nearly 500 process models. This is a small number compared to the size of process model repositories maintained in multi-national companies, which typically contain several thousand models [1]. The SAP reference model repository, which we use in this paper for experimental purposes, contains 604 process models.

The management of large process model repositories requires effective search techniques. For example, before adding a new process model to a repository, one needs to check that a similar model does not already exist in order to prevent duplication. Similarly, in the context of company mergers, process analysts need to identify common or similar business processes between the merged companies in order to analyze their overlap and to identify areas for consolidation. These tasks require users to retrieve process models based on their similarity with respect to a given "search model". We use the term *process model similarity query* to refer to such search queries over process model repositories.

One may argue that traditional (text-based) search engines can be used to index and to search business process model repositories. However, text-based search engines are based on keyword search and text similarity.

* Corresponding author.
E-mail addresses: r.m.dijkman@tue.nl (R. Dijkman), marlon.dumas@ut.ee (M. Dumas), b.f.v.dongen@tue.nl (B. van Dongen), reinak@ut.ee (R. Käärik), contact@mendling.com (J. Mendling).

0306-4379/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2010.09.006

They are clearly useful in situations where a user is looking for a model that contains a task with a certain keyword in its label. On the other hand, it is unclear in how far search engines are appropriate for process model similarity queries, since they do not take into account the structure and behavioral semantics of process models. What is needed in this research area is a comparative evaluation of different approaches to this problem.

This paper studies three classes of similarity metrics designed to answer process model similarity queries. Within each class we study some variations as explained further on in the paper. The three classes of metrics are derived from the increasing levels of "semantic richness" at which business process models can be considered: we can consider individual tasks, tasks and their relations and the behavior of an entire process as it is induced by tasks and relations. Consequently, the first class of metrics exploits the fact that process models are composed of labeled nodes. These metrics start by calculating an optimal matching between the nodes in the process models by comparing their labels. Based on this matching, a similarity score is calculated taking into account the overall size of the models. The second class of metrics is structural. It is based on the observation that nodes in process models with their relations constitute a mathematical graph. Based on that observation it uses existing techniques for graph comparison based on graph–edit distance [2], which is commonly used in information retrieval. The third class of metrics is behavioral, in the sense that it takes into account the causal relations between tasks in a process model. These causal relations are represented in the form of a *causal footprint* [3].

The paper is an extension of our earlier works [4–6] in which we introduced structural and behavioral similarity notions along with initial evaluations. In this paper, we provide a comparative evaluation of these two notions of process model similarity with node match similarity. We present an extensive evaluation using a text-based search engine as a baseline for comparison in two dimensions. First, the evaluation is done using classical measures of quality for ranked retrieval results, including mean average precision and first-10 precision. In this evaluation, we compare the proposed similarity metrics with a text-based search engine. Second, we give an account of performance evaluation, which suggests that all proposed approaches are applicable in the envisaged use cases.

The remainder of the paper is structured as follows. Section 2 presents the notation used to represent business process models. Sections 3, 4 and 5 present the label-based, structure-based and behavior-based similarity metrics, respectively. Section 6 presents the experimental evaluation. Finally, Sections 7 and 8 present related work and conclusions.

2. Preliminaries

This section introduces notations and notions used in the rest of the paper. Firstly, the section introduces the notion of a business process graph (BPG), which we will use as the formalism on which the similarity metrics are defined. Secondly, it introduces the notion of causal footprint [3], which provides an abstract representation of the behavior of a business process model. Causal footprints will be used in Section 5 in order to define the behavioral similarity metrics. Thirdly, the section defines two similarity metrics for comparing pairs of labels. The process model similarity metrics studied in the paper rely on these similarity metrics in order to compare process model elements.

2.1. Business process graphs

Numerous notations compete in the business process modeling space, including UML Activity Diagrams, the Business Process Modeling Notation (BPMN), Event-driven Process Chains (EPCs), Workflow nets, and the Business Process Execution Language (BPEL)—the latter one being intended for executable specification rather than modeling. However, our aim is to define similarity metrics that can be applied to all these different notations. To achieve this level of generality, we define similarity metrics based on the so-called business process graphs rather than on a specific notation. In this way, we also enable measuring the similarity of business processes modeled in different notations.

A business process graph (BPG) is simply a graph that captures node and edge types of different notations as attributes. This definition is based on the observation that, although many notations exist for modeling business processes, most of them are graph-based. Even the so-called structured modeling languages, such as BPEL, can be trivially mapped to a graph-based notation as discussed in [7]. Furthermore, there is a considerable overlap between existing languages [8]: all are based on activity nodes, and nodes with the same routing behavior can be annotated with the same attributes, e.g. BPMN AND-gateways and UML Activity Diagram forks. Finally, there are transformations available for all relevant business process modeling languages to Petri nets [9]. For many languages these transformations are complete while only a few of the constructs cannot be directly expressed as, for instance, OR-joins. Yet, the behavioural abstraction that we will use later, namely causal footprints, is even capable of representing OR-joins. Therefore, if we define the similarity metrics on BPGs, they can be used for all graph-based notations and even between different graph-based notations.

Definition 1 (*BPG*). Let T be a set of types and Ω be a set of text labels. A BPG is a tuple $(N,E,\tau,\lambda,\alpha)$, in which:

- N is a finite set of nodes;
- $E : N \times N$ is a finite set of edges;
- $\tau : (N \cup E) \rightarrow T$ associates nodes and edges with a types;
- $\lambda : (N \cup E) \rightarrow \Omega$ associates nodes and edges with labels; and
- $\alpha : (N \cup E) \rightarrow (T \rightarrow \Omega)$ associates nodes and edges with attributes, where an attribute always is a combination of a type and a label.

Fig. 1 shows an example of a business process model in the BPMN notation with the corresponding BPG. In

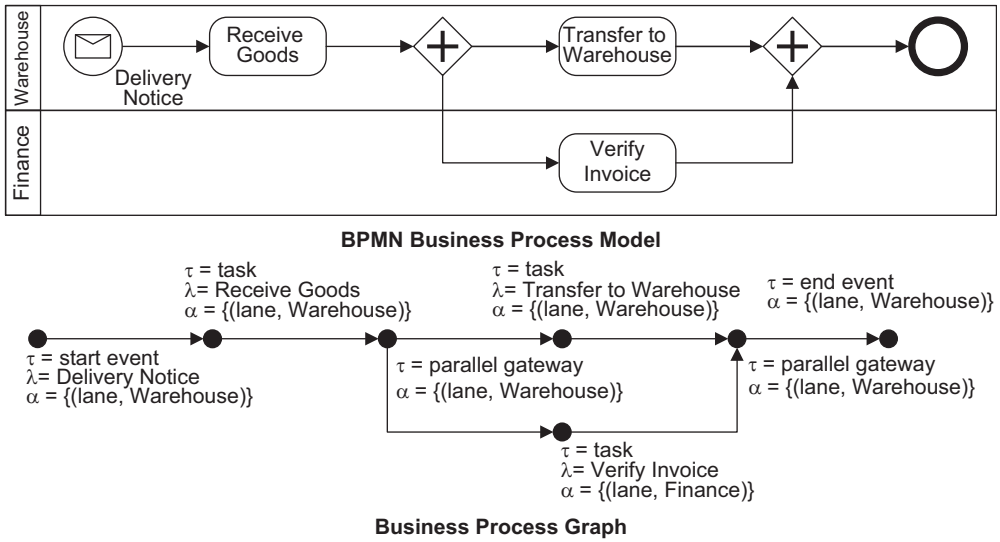

Fig. 1. A business process model and its business process graph.

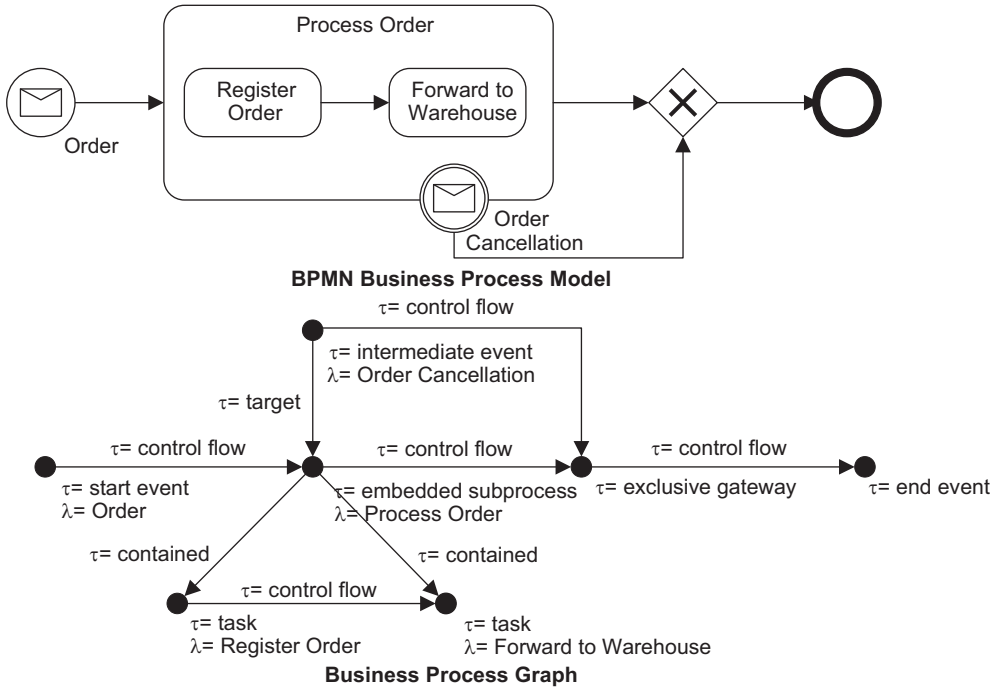

Fig. 2. A business process model and its business process graph with typed edges.

process modeling notations different types of nodes and edges are identified by different notational elements. For example, in BPMN events are identified by circles, tasks by rounded rectangles, gateways by diamonds, control flows by arrows and message flows by dashed arrows. In BPGs the type of a node is identified by the function τ. In addition to that process modeling notations allow various attributes to be associated with a node. For example, in BPMN a task can be drawn inside a lane, identifying the role that performs the task, and a multiple instance task can have an attribute that defines the number of instances. In BPGs attributes are associated to a node through the function α.

Fig. 2 shows another example of a business process model and the corresponding BPG. This example demonstrates the use of "typed edges" to represent relations other than the control flow relation, that can exist between nodes. In process modeling notations different node relations can be represented in a number of different ways. For example, as illustrated in Fig. 2, the

BPMN notation allows for the use of containment to represent the relation between a subprocess and the activities that are part of that subprocess. In addition to that it allows for the use of events on the boundary of an activity to represent that the event can interrupt the activity. In a BPG such relations cannot be represented in this manner, because a BPG only contains nodes and edges. Therefore, we use edges of different types to represent different relations between nodes. In Fig. 2 the relation between a subprocess and its parts is represented by edges typed "contained" and the relation between an interrupting event and the activity that it can interrupt is represented by an edge typed "target".

Below we define our similarity metrics for business process graphs with an arbitrary set of types T, such that the similarity metrics will work for any graph-based notation. To also enable comparison between different graph-based notations the set of types T must be standardized. For example, to enable comparison between BPMN, UML Activity Diagrams and EPCs, the set of types T should contain a type "task" and the BPMN "task" type, the UML Activity Diagram "Activity" type and the EPC "Function" type should be mapped to this type. Notation specific elements must be mapped to more general types in a BPG *before* the similarity metrics can be applied to models in different notations. We consider this mapping out of the scope of this paper. However, we are pursuing related research in this direction [10].

In the remainder of this paper we will use the notions of path and typed path to discuss the relations between nodes.

Definition 2 (*Paths and typed paths*). Let $(N,E,\tau,\lambda,\alpha)$ be a BPG and $a,b \in N$ be two nodes. A *path* $a \hookrightarrow b$ refers to the existence of a sequence of nodes $n_1,\ldots,n_k \in N$ with $a=n_1$ and $b=n_k$ such that for all $i \in 1,\ldots,k$ holds: (n_1,n_2), $(n_2,n_3),\ldots,(n_{k-1},n_k) \in E$. This includes the empty path (i.e.: $a \hookrightarrow a$ if $(a,a) \in E$). Let $ts \subseteq T$ be a set of types. A path containing only nodes $n_2,\ldots,n_{k-1}$ that are of type $t \in ts$, denoted $a \overset{ts}{\hookrightarrow} b$, is called a *ts* typed path. This includes the empty typed path (i.e.: $a \overset{ts}{\hookrightarrow} a$ if $(a,a) \in E$).

2.2. Causal footprints

A *causality graph* is a set of activities and conditions on when those activities can occur. Its intended use is as a formal semantics that approximates the behavior of a business process, in which case we also refer to it as the *causal footprint* of that process. One of the advantages that causal footprints have over other formal semantics (e.g. semantics in terms of a state-space or a trace-set) is that causal footprints remain relatively small, while other formal representations are combinatorially large or even infinite when used to represent the behavior of business process models [11]. This makes causal footprints more practical for use in algorithms for which a response is required in a matter of milliseconds (i.e. search algorithms). Note, however, that a causal footprint is an approximation of the behavior of a business process, making it suitable only for use in algorithms that do not require an exact behavioral semantics.

A causality graph represents behavior between a set of activities by means of two relationships, namely look-back and look-ahead links. For a *look-ahead link* from an activity to a (non-empty) set of activities, we say that the execution of that activity leads to the execution of at least one of the activities in the set. I.e. if (a,B) is a look-ahead link, then any execution of a is directly or indirectly followed by the execution of some $b \in B$. Furthermore, for a *look-back link* from a (non-empty) set of activities to an activity, we say that the execution of the activity is preceded by the execution of at least one of the activities in the set. I.e. if (A,b) is a look-back link, then any execution of b is directly or indirectly preceded by the execution of some $a \in A$.

Definition 3 (*Causality graph*). A causality graph is a tuple (A,L_{lb},L_{la}), in which:

- A is a finite set of activities.
- $L_{lb} \subseteq (\mathcal{P}(A) \times A)$ is a set of look-back links.[1]
- $L_{la} \subseteq (A \times \mathcal{P}(A))$ is a set of look-ahead links.

A causality graph is a causal footprint of a business process if and only if it is consistent with the behavior of that process. In the definition below we consider the behavior as a set of traces A^* over an alphabet A. This set of traces is not needed for the computation of the causal footprint. The causal footprint merely has to be consistent with that semantics. We refer to [3] for an algorithm to compute a causal footprint of an EPC or Petri-net. The causal footprint of a BPMN model can be computed indirectly using the Petri-net based semantics of BPMN [12].

Definition 4 (*Causal footprint*). Let $(N,E,\tau,\lambda,\alpha)$ be a BPG and ts be the set of types for which we build a causal footprint. Then $A=\{n|n \in N,\tau(n) \in ts\}$ is the set of nodes for which we build the causal footprint. Furthermore, let $G=(A,L_{lb},L_{la})$ be a causality graph over the set of nodes A, and $W \subseteq A^*$ be the set of possible orders in which the nodes from A can be performed. G is a causal footprint of the BPG if and only if:

1. For all $(a,B) \in L_{la}$ holds that for each $\sigma \in W$ with $n=|\sigma|$, such that there is a $0 \le i \le n-1$ with $\sigma[i]=a$, there is a $j : i<j \le n-1$, such that $\sigma[j] \in B$.
2. For all $(A,b) \in L_{lb}$ holds that for each $\sigma \in W$ with $n=|\sigma|$, such that there is a $0 \le i \le n-1$ with $\sigma[i]=b$, there is a $j : 0 \le j < i$, such that $\sigma[j] \in A$.

Note that the definition only develops a causal footprint for a subset of the nodes in a BPG, because typically the behavioral semantics (i.e. the set of possible orders in which the nodes are performed) is only defined on certain types of nodes. For example, in BPMN the set of

[1] With $\mathcal{P}(A)$, we denote the powerset of A, where $\emptyset \notin \mathcal{P}(A)$.

possible orders could be defined in terms of tasks, or in terms of tasks and events.

As an example, a possible causal footprint for the business process model from Fig. 1, focusing only on tasks, has the look-ahead link ("Receive Goods", {"Verify Invoice", "Transfer to Warehouse"}) and look-back links ({"Receive Goods"}, "Verify Invoice") and ({"Receive Goods"}, "Transfer to Warehouse"). This example illustrates that causal footprints are an approximation of the behavior of a business process, because there are multiple business processes that have the same causal footprint (for example, the business process that can be derived from Fig. 1 by transforming the parallel block in a choice block). Also, there are multiple possible causal footprints for the same business process model.

2.3. Similarity of process model elements

When comparing business process models it is not realistic to assume that their elements (nodes) are only equivalent if they have exactly the same label. Fig. 3 is an example in point: tasks "Customer inquiry processing" and "Client inquiry query processing" would be considered as practically identical by a process modeler, although they have different labels. Therefore, as a basis for measuring the similarity between business process models, we must be able to measure the similarity between their elements. We consider five ways of measuring similarity between elements of different process models (see Fig. 4):

1. Syntactic similarity, where we consider the syntax of labels.
2. Semantic similarity, where we look at the semantics of the words within the labels.
3. Attribute similarity, where we look at the attribute values.
4. Type similarity, where we look at the node types.
5. Contextual similarity, where we do not only consider the similarity of two nodes, but also the context in which these nodes occur.

All these metrics (as described below) result in a similarity score between 0 and 1, where 0 indicates no similarity and 1 indicates identical elements. Hence, it is trivial to combine all metrics to obtain a weighted similarity score.

We experimented with other metrics for determining the similarity of process model elements, inspired by the work of Ehrig et al. [13] and we also experimented with different parameters for the metrics presented below. However, we obtained the best results for the metrics and parameters explained below, based on an evaluation of different metrics to determine the similarity between 210 pairs of process model elements [14].

2.3.1. Syntactic Similarity

Given two labels (e.g. the labels of two nodes or the labels of two node attributes), the syntactic similarity metric returns the degree of similarity as measured by the string–edit distance. The string–edit distance [15] is the number of atomic string operations necessary to get from one string to another. These atomic string operations include: removing a character, inserting a character or substituting a character for another.

Definition 5 (*Syntactic similarity*). Let $l,l_1,l_2 \in \Omega$ be text labels. Furthermore, let $|l|$ be the length of a text label l and $\mathrm{ed}(l_1,l_2)$ be the edit distance of text labels l_1 and l_2. We define the *syntactic similarity* of text labels l_1 and l_2, denoted $\mathrm{syn}(l_1, l_2)$, as follows:

$$1-\frac{\mathrm{ed}(l_1,l_2)}{\max(|l_1|,|l_2|)}$$

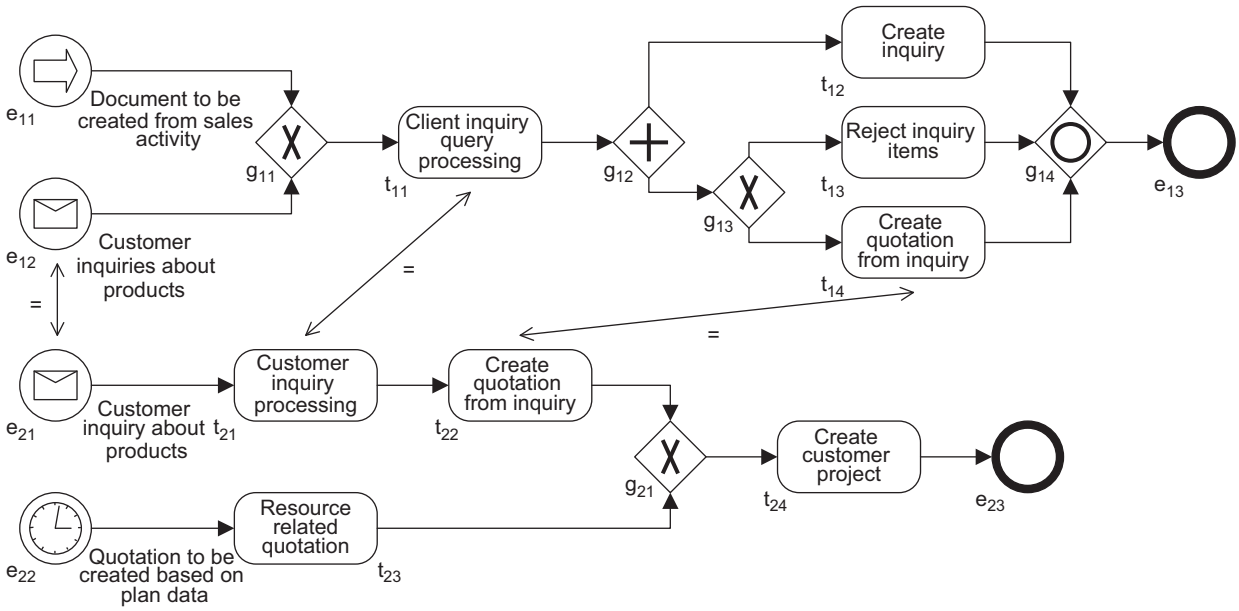

Fig. 3. Two customer inquiry processes.

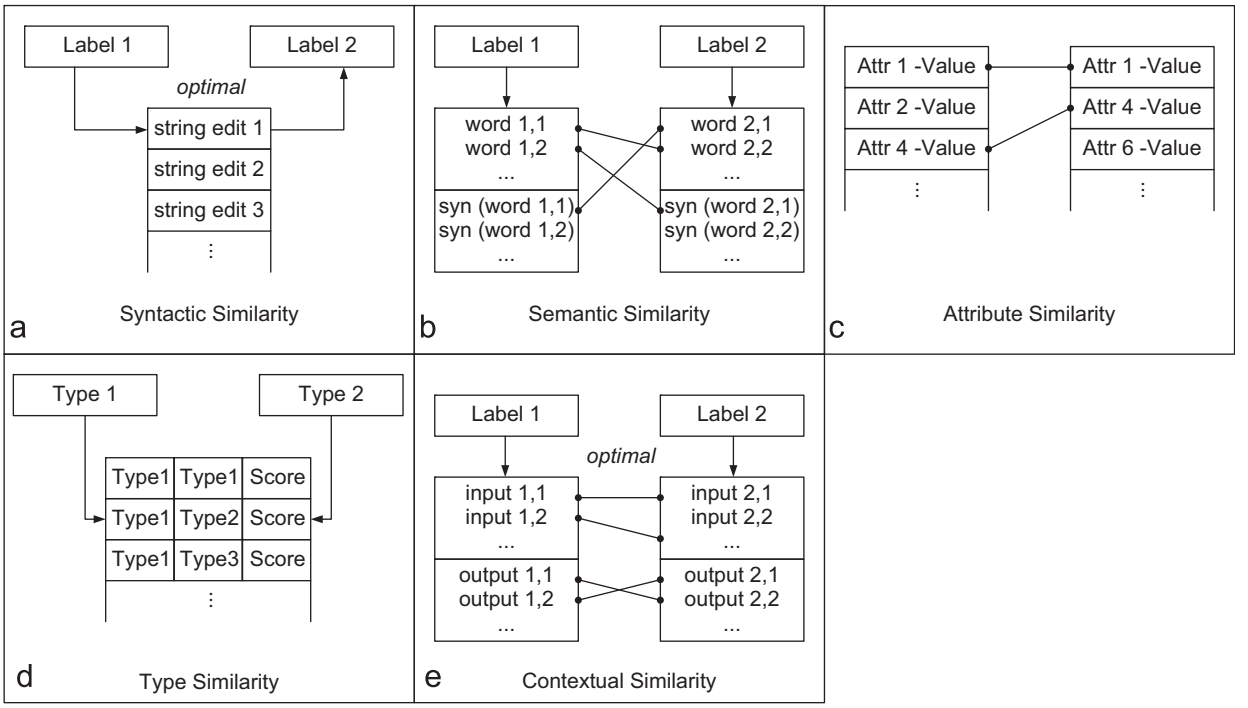

Fig. 4. Overview of different approaches to similarity of process model elements.

Let $(N_1,E_1,\tau_1,\lambda_1,\alpha_1)$ and $(N_2,E_2,\tau_2,\lambda_2,\alpha_2)$ be two BPGs and let $n_1 \in N_1$ and $n_2 \in N_2$ be two nodes from those BPGs. We define the *syntactic similarity* of nodes n_1 and n_2 as follows:

$$\text{Simsyn}(n_1,n_2) = \text{syn}(\lambda_1(n_1),\lambda_2(n_2))$$

For example, the syntactic similarity between the events e_{12} and e_{21} from Fig. 3 with labels "Customer inquiry about product" and "Customer inquiries about product" is $1-\frac{3}{30}=0.90$, because the edit distance is 3 ("inquiries" becomes "inquiry" by substituting the "y" with a "i" and inserting an "e" and an "s"). For comparing labels we disregard special symbols, such as newline, brackets and quotes and we change all characters to lower-case.

2.3.2. Semantic similarity

Given two labels, their semantic similarity score is the degree of similarity, based on equivalence between the words they consist of. Hence, the semantic similarity score is defined as follows.

Definition 6 (*Semantic similarity*). Let $l,l_1,l_2 \in \Omega$ be text labels, W be the set of all words, $w : \Omega \to \mathcal{P}(W)$ be a function that separates a label into a set of words and $s : W \to \mathcal{P}(W)$ be a function returns the set of synonyms for a given word (based on a dictionary lookup). Furthermore, let $w_1 = w(l_1)$ and $w_2 = w(l_2)$ and let wi and ws be the weights that we associate with identical words and synonymous words, respectively. We define the *semantic similarity* of labels l_1 and l_2, denoted $\text{sem}(l_1,l_2)$, as follows:

$$\frac{2 \cdot wi \cdot |w_1 \cap w_2| + ws \cdot (|s(w_1,w_2)| + |s(w_2,w_1)|)}{|w_1| + |w_2|}$$

where $s(w_1,w_2)$ is the set of synonyms of w_1 that appear in w_2.

$$s(w_1,w_2) = \bigcup_{w \in w_1 - w_2} s(w) \cap (w_2,w_1)$$

Let $(N_1,E_1,\tau_1,\lambda_1,\alpha_1)$ and $(N_2,E_2,\tau_2,\lambda_2,\alpha_2)$ be two BPGs and let $n_1 \in N_1$ and $n_2 \in N_2$ be two nodes from those BPGs. We define the *semantic similarity* of nodes n_1 and n_2 as follows:

$$\text{Simsem}(n_1,n_2) = \text{sem}(\lambda_1(n_1),\lambda_2(n_2))$$

For example, suppose we assign $wi=1.0$ and $ws=0.75$ and consider the tasks t_{11} and t_{21} from Fig. 3 with labels "Client inquiry query processing" and "Customer inquiry processing". These labels consist of the collections of words w_1=["Client", "inquiry", "query", "processing"] and w_2= ["Customer","inquiry","processing"], respectively. We only need to consider a synonym mapping between $w_1 \backslash w_2 =$ [" Client" ," query"] and $w_2 \backslash w_1 =$ [" Customer"]. We consider "Customer" and "Client" synonymous and "Customer" and "query" not synonymous. Therefore, the semantic similarity between w_1 and w_2 equals $\text{sem}(w_1,w_2) = (1.0 \cdot 2 + 0.75 \cdot (1+0))/4 \approx 0.69$.

When determining equivalence between words, we disregard special symbols, and we change all characters to lower-case. Furthermore, we skip frequently occurring words, such as "a", "an" and "for" and we *stem* words using Porter's stemming algorithm [16]. Stemming reduces words to their stem form. For example, "stemming", "stemmed" and "stemmer" all become "stem".

In previous work [14] we established experimentally that $wi=1.0$ and $ws=0.75$ are adequate values. For this, we manually compared 210 function pairs from the SAP

Reference Model. For each pair, we determined if their labels matched according to our own judgement. We then calculated the semantic similarity score using different synonymy weight factors (0, 0.25, 0.5, 0.75 and 1). For each possible synonymy weight factor, we sorted the pairs according to their calculated similarity score, and checked if those pairs that we had manually identified as being "semantically equivalent" appeared at the top of the list. Using the synonymy weight factor of 0.75, led to 90% of the pairs that we manually tagged as semantically equivalent appearing at the top of the list.

2.3.3. Attribute similarity

Given two nodes, we can determine their similarity of their attribute values. The similarity of the attributes then is defined as the average of the similarity of attributes of the same type.

Definition 7 (*Attribute similarity*). Let $(N_1,E_1,\tau_1,\lambda_1,\alpha_1)$ and $(N_2,E_2,\tau_2,\lambda_2,\alpha_2)$ be two BPGs and let $n_1 \in N_1$ and $n_2 \in N_2$ be two nodes from those BPGs. Furthermore, let s be one of the functions syn or sem. We define the *attribute similarity* of nodes n_1 and n_2 as follows:

$$\mathrm{Simattr}(n_1,n_2) = \underset{\substack{(t_1,l_1)\in\alpha_1(n_1),\\ (t_2,l_2)\in\alpha_2(n_2),t_1=t_2}}{\mathrm{AVG}} s(l_1,l_2)$$

This similarity metric will most likely not be used by itself, because it ignores the similarity of node labels, while similarity of node labels is typically a strong indication that the nodes themselves are similar. However, it can easily be combined with the syntactic or semantic similarity metric.

2.3.4. Type similarity

The similarity of two nodes largely depends on the similarity of their types. In particular, it may be desirable to only consider the similarity of nodes in case they are of the same type. Alternatively, the similarity of nodes that are of a related type can also be considered, potentially to a lesser degree than the similarity of nodes that are of the same type. For example, the similarity of a node of type "send message" and a node of type "receive message" can be considered to some specified degree, such that the similarity of the node with type "send message" and label "order" and the node "receive message" and label "order" is 0.7. We define the following function to determine type similarity of nodes.

Definition 8 (*Type similarity*). Let $(N_1,E_1,\tau_1,\lambda_1,\alpha_1)$ and $(N_2,E_2,\tau_2,\lambda_2,\alpha_2)$ be two BPGs and let $n_1 \in N_1$ and $n_2 \in N_2$ be two nodes from those BPGs. Furthermore, let $\mathrm{typ} : T \times T \rightarrow [0 \ldots 1]$ be the function that assigns similarity scores to pairs of types. We define the *type similarity* of the types of nodes n_1 and n_2 as follows:

$$\mathrm{Simtyp}(n_1,n_2) = \mathrm{typ}(\tau_1(n_1),\tau_2(n_2))$$

The function typ that defines the similarity of types has to be predefined as desired. The simplest function is the function that only considers the potential similarity of nodes in case they are of the same type:

$$\mathrm{typ}(t_1,t_2) = \begin{cases} 1 & \text{if } t_1 = t_2 \\ 0 & \text{otherwise} \end{cases}$$

Like the attribute similarity function, is is not likely that this metric will be used by itself. Instead, it can be used in combination with other metrics, such that nodes with differing types automatically receive a lower (or zero) similarity than nodes with the same type.

2.3.5. Contextual similarity

The metrics defined above focus on the similarity of two process model elements. We now define a fifth similarity metric that, when determining the similarity of two model elements, also takes the model elements that precede and succeed them into account. Such a similarity metric is especially useful in notations in which "active" and "passive" model elements are strictly alternating (i.e.: model elements that appear or do not appear in the set of execution traces). This includes the EPC and the Petri net notation. In the EPC notation functions and events are strictly alternating and in the Petri net notation transitions and places are strictly alternating.

We refer to preceding model elements as the *input context* and to succeeding model elements as the *output context* of another model element. When determining the preceeding or succeeding model elements we may choose to ignore certain types of modeling elements, such as gateways.

Definition 9 (*Input and output context*). Let $(N,E,\tau,\lambda,\alpha)$ be a BPG and let *ts* be the set of types of contextual elements that should be ignored. For a node $n \in N$, we define the input context $n^{in} = \{n' \in N | n' \overset{ts}{\hookrightarrow} n\}$ and the output context $n^{out} = \{n' \in N | n \overset{ts}{\hookrightarrow} n'\}$

To determine the contextual similarity between elements of a business process model, we need to establish the equivalence between elements in their input contexts and the equivalence between elements in their output contexts. We establish those equivalences by computing the *equivalence mapping* as defined below. In this paper we always assume that an element can be mapped to at most one other element. We do that to prevent explosion of possible relations between elements and therewith computational explosion of algorithms to compute the metrics further on in this paper.

Definition 10 (*Equivalence mapping*). Let $(N_1,E_1,\tau_1,\lambda_1,\alpha_1)$ and $(N_2,E_2,\tau_2,\lambda_2,\alpha_2)$ be two BPGs. Furthermore, let $\mathrm{Sim} : N_1 \times N_2 \rightarrow [0..1]$ be a similarity function. A partial injective mapping $M_{\mathrm{Sim}} : N_1 \nrightarrow N_2$ is an *equivalence mapping*, if and only if for all $n_1 \in N_1$ and $n_2 \in N_2$: $(n_1,n_2) \in M$ implies that $\mathrm{Sim}(n_1,n_2) > 0$.

An *optimal equivalence mapping* $M_s^{opt} : N_1 \nrightarrow N_2$ is an equivalence mapping, such that for all other equivalence mappings M holds that $\sum_{(n_1,n_2)\in M_{\mathrm{Sim}}^{opt}} \mathrm{Sim}(n_1,n_2) \geq \sum_{(n_1,n_2)\in M_{\mathrm{Sim}}} \mathrm{Sim}(n_1,n_2)$.

For example, in Fig. 3 we can develop an equivalence mapping between the sets $\{e_{12}\}$ and $\{e_{21},e_{22}\}$, using syntactic similarity (*syn*) as a similarity function. M_{syn}=

$\{(e_{12},e_{22})\}$ is a possible equivalence mapping, because $\text{syn}(e_{12},e_{22}) \approx 0.24$. $M^{opt}_{syn}=\{(e_{12},e_{21})\}$ is the optimal equivalence mapping, because $\text{syn}(e_{12},e_{21})=0.90$. The only other possible mapping is the empty mapping.

Now, we use the concept of equivalence mappings to determine the contextual similarity between nodes.

Definition 11 (*Contextual similarity*). Let $(N_1,E_1,\tau_1,\lambda_1,\alpha_1)$ and $(N_2,E_2,\tau_2,\lambda_2,\alpha_2)$ be two BPGs and let $n_1 \in N_1$ and $n_2 \in N_2$ be two nodes from those BPGs. Furthermore, let Sim be one of the similarity functions from Sections 2.3.1, 2.3.2, or 2.3.3 and let *ts* be the set of types of contextual elements that should be ignored. Furthermore, let $M^{optin}_{\text{Sim}} : n^{in}_1 \nrightarrow n^{in}_2$ and $M^{optout}_{\text{Sim}} : n^{out}_1 \nrightarrow n^{out}_2$ be two optimal equivalence mappings between the input and output contexts of n_1 and n_2, which ignore types from *ts*. We define the contextual similarity that ignores types from *ts* as follows:

$$\text{Simcon}(n_1,n_2) = \frac{|M^{optin}_{\text{Sim}}|}{2 \cdot \sqrt{|n^{in}_1|}} \cdot \sqrt{|n^{in}_2|} + \frac{|M^{optout}_{\text{Sim}}|}{2 \cdot \sqrt{|n^{out}_1|}} \cdot \sqrt{|n^{out}_2|}$$

In the remainder of this paper, we use $\text{Sim}(n_1,n_2)$ to denote the similarity value between two elements of a model. Any of the symmetric similarity functions above (Simsyn, Simsem, Simattr, Simtyp or Simcon) can be substituted for this, as well as any weighted combination thereof if the sum of weights is 1.

3. Node matching similarity

The first similarity measure we study, namely *node matching similarity*, is based on pairwise comparisons of node labels or attributes. It is obtained by calculating an optimal equivalence mapping between the nodes of the two process models being compared (see illustration in Fig. 5). The node matching similarity score is the sum of the label similarity scores of the matched pairs of nodes. To obtain a score between 0 and 1, we divide the sum by the total number of nodes.

Definition 12 (*Node matching similarity*). Let $B_1=(N_1,E_1,\tau_1,\lambda_1,\alpha_1)$ and $B_2=(N_2,E_2,\tau_2,\lambda_2,\alpha_2)$ be two BPGs, let Sim be a function that assigns a similarity score to a pair of nodes and let *ts* be a set of types of nodes that should be ignored. Furthermore, let $M^{opt}_{\text{Sim}} : (N_1 \nrightarrow N_2)$ be an optimal equivalence mapping derived from Sim, which ignores types from *ts*. The *node matching similarity* between B_1 and B_2 is

$$\text{simnm}(B_1,B_2) = \frac{2 \cdot \Sigma_{(n,m)\in M^{opt}_{\text{Sim}}} \text{Sim}(n,m)}{|\{n|n \in N_1,\tau_1(n)\notin ts\}| + |\{n|n \in N_2,\tau_2(n)\notin ts\}|}$$

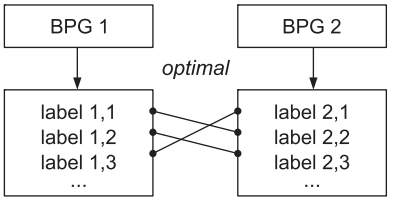

Fig. 5. Illustration of node matching similarity calculation.

The node matching similarity metrics is parameterized by the similarity metrics used to compare pairs of nodes. We can use the syntactic, semantic, attribute, type or context similarity notions defined in Section 2.3, or a weighted average of them. We further parameterize the node matching similarity metrics with a threshold between 0 and 1. When calculating an optimal equivalence mapping, we only allow two nodes to be included in the equivalence mapping if their similarity is above the threshold. With respect to Definition 10, this means that instead of enforcing that $\text{Sim}(n_1,n_2) > 0$, we enforce that $\text{Sim}(n_1,n_2) \geq \text{threshold}$.

As an example, consider the process models from Fig. 3. The optimal equivalence mapping between these models, ignoring gateway types, is denoted by the two-way arrows with the = symbol on them. Assuming that we use syntactic equivalence (Simsyn) to determine the similarity between nodes, and that we use a threshold of 0.5, the similarity score of the elements included in the equivalence mapping is: $\text{Simsyn}(e_{12},e_{21})=0.90$, $\text{Simsyn}(t_{11},t_{21}) \approx 0.58$ and $\text{Simsyn}(t_{14},t_{22})=1.00$. The remaining elements are not included in the equivalence mapping because the syntactic similarity score between all other possible pairs of elements in this example is less than 0.5. Hence, the node matching similarity between these two models is

$$\frac{2 \cdot \Sigma_{(n,m)\in M^{opt}_{\text{Simsyn}}} \text{Simsyn}(n,m)}{|\{n|n \in N_1,\tau_1(n)\notin ts\}| + |\{n|n \in N_2,\tau_2(n)\notin ts\}|}$$

$$= \frac{2 \cdot (0.90+0.58+1.00)}{6+4}$$

4. Structural similarity

The second similarity metric we study is a similarity metric over the structure of a business process model. We define that metric based on the graph–edit distance [2] of business process graphs (see Fig. 6). The graph–edit distance between two graphs is the minimal number of graph–edit operations that is necessary to get from one graph to the other. Different graph–edit operations can be taken into account. We take into account: node deletion or insertion, node substitution (a node is a graph is mapped to a node in the other graph with a different label), and edge deletion or insertion.

Like the node matching similarity, graph–edit distance is obtained by first computing a mapping between nodes

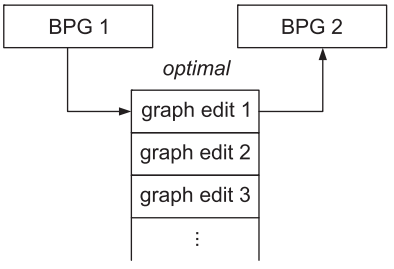

Fig. 6. Illustration of structural similarity calculation.

and subsequently computing the optimal graph–edit distance. This score is computed as follows.

- We consider two mapped nodes "substituted". Their distance is one minus the similarity of their labels, because this value represents the effort necessary to substitute one node (or rather its label) for the other.
- We consider an unmapped node either deleted or inserted.
- If there is an edge between two nodes in one graph, then we consider that edge to exist in the other graph if and only if the nodes are mapped to nodes in the other graph and there is an edge between the mapped nodes. Otherwise, we consider the edge deleted or inserted.

Definition 13 (*Graph–edit distance*). Let $B_1=(N_1,E_1,\tau_1,\lambda_1,\alpha_1)$ and $B_2=(N_2,E_2,\tau_2,\lambda_2,\alpha_2)$ be two BPGs and let Sim be one of the similarity metrics from Section 2.3. Furthermore, let $M:(N_1 \nrightarrow N_2)$ be a partial injective mapping.

Let $n \in N_1 \cup N_2$ be a node. n is substituted if and only if $n \in \text{dom}(M)$ or $n \in \text{cod}(M)$. sb is the set of all substituted nodes. n is inserted or deleted if and only if it is not substituted. sn is the set of all inserted and deleted nodes.

Let $(n,m) \in E_1$ be an edge. (n, m) is inserted in or deleted from B_1 if and only if there do not exist mappings $(n,n') \in M$ and $(m,m') \in M$ and edge $(n',m') \in E_2$. Edges that are inserted in or deleted from B_2 are defined similarly. se is the set of all inserted or deleted edges.

The distance induced by the mapping is defined as

$$|sn|+|se|+2\cdot\Sigma_{(n,m)\in M}1-(\text{Sim}(n,m))$$

The graph–edit distance is the minimal possible distance induced by a mapping between the two processes.

As an example, consider the process models from Fig. 3. Assuming that we use syntactic similarity (Simsyn) to determine the similarity between nodes, the distance of the mapping that is displayed in the figure is: $13+19+2\cdot(1-0.90+1-0.58+1-1.00)\approx 33{,}04$.

The graph–edit distance similarity is computed as one minus the average of the fraction of inserted or deleted nodes, the fraction of inserted of deleted edges and the average distance of substituted nodes.

Definition 14 (*Graph–edit distance similarity*). Let $B_1=(N_1,E_1,\tau_1,\lambda_1,\alpha_1)$ and $B_2=(N_2,E_2,\tau_2,\lambda_2,\alpha_2)$ be two BPGs and let Sim be one of the similarity metrics from Section 2.3.

Furthermore, let $M:(N_1 \nrightarrow N_2)$ be the partial injective mapping that induces the graph–edit distance between the two processes and let sn and se be defined as in Definition 13. We define the *graph–edit distance similarity* as

$$\text{simged}(B_1,B_2)=1-avg(\text{snv},\text{sev},\text{sbv})$$

where

$$\text{snv}=\frac{|\text{sn}|}{|N_1|+|N_2|}$$

$$\text{sev}=\frac{|\text{se}|}{|E_1|+|E_2|}$$

$$\text{sbv}=\frac{2\cdot\Sigma_{(n,m)\in M}1-\text{Sim}(n,m)}{|N_1|+|N_2|-|\text{sn}|}$$

Variations of this metric are possible. The user of the technique can choose the particular variation of the technique that must be used. One variation is to use the weighted average, instead of the plain average, of the fractions of skipped nodes, substituted nodes and skipped edges. If this variation is chosen, the user must choose the appropriate weights. Another variation is to ignore certain types of nodes. We ignore nodes by removing them from the BPG and replacing paths through ignored nodes by direct edges.

Definition 15 (*Node abstraction*). Let $B=(N,E,\tau,\lambda,\alpha)$ be a BPG, let Ω be the set of all possible labels, T be the set of all types and let ts be the set of types to ignore. The BPG in which the nodes of type $t \in ts$ are ignored is the BPG $B'=(N',E',\tau',\lambda',\alpha')$, where:

- $N'=\{n|n \in N,\tau(n)\notin ts\}$;
- $E'=(E\cap(N'\times N'))\cup\{(n,m)|n,m\in N,n\overset{ts}{\hookrightarrow}m\}$;
- $\tau'=\tau\cap((N'\cup E')\times T)$;
- $\lambda'=\lambda\cap((N'\cup E')\times\Omega)$; and
- $\alpha'=\alpha\cap((N'\cup E')\times(T\times\Omega))$.

For example, when using graph–edit distance similarity in Fig. 3, all edges are inserted or deleted, leading to the maximal edit distance with respect to edges. However, there are indirect edges from e_{12} to t_{11} and from t_{11} to t_{14} via gateway nodes. Therefore, one could argue that the edit distance is too high (and therefore the edit distance similarity too low) and that insertion and deletion of gateway nodes can lead to incorrect similarity measurements. This issue can be addressed by ignoring all gateway nodes, but of course that would mean that gateway nodes are not considered in the similarity metric at all.

5. Behavioral similarity

The third similarity metric we study takes into account the behavior of a process model. The benefit of using behavioral similarity over structural similarity is illustrated by the point raised at the end of Section 4, namely that indirect edges via the inserted or deleted nodes are not considered in structural similarity, while they are relevant. In behavioral similarity indirect relations *are* considered (see Fig. 7). For example in the behavior of the model from Fig. 3 there is a direct relation between event e_{12} and task t_{11} (i.e. e_{12} is in the look-back link of t_{11}), while there is only an indirect relation in their structure, which is ignored in structural similarity and leads to a lower structural similarity score.

We compute the behavioral similarity of two process models by computing their distance in the document vector space constructed from their causal footprints. Fig. 8 illustrates this idea with a simple example. The

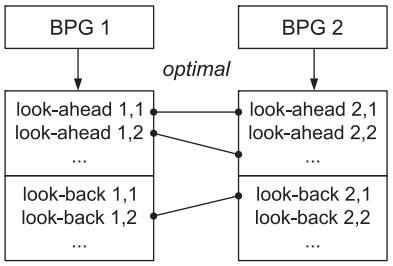

Fig. 7. Illustration of behavioral similarity calculation.

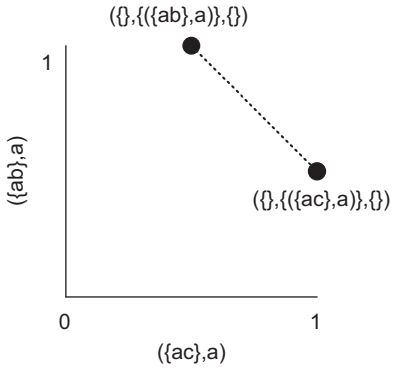

Fig. 8. Example vector space for two causal footprints.

figure shows a vector space for two causal footprints, which both consist of a single look-back link. The vector space has two axes, one for each look-back link. The two causal footprints can be positioned inside this space (represented by the dots) and subsequently their distance can be determined (represented by the dashed line). The vector space consists of all look-back and look-ahead links from both models. Consequently, if a node appears in one model but not in the other, the similarity of the causal footprints is lowered, because the look-back and look-ahead links for the node receive a score of 0 in the model in which the node does not appear (and a 1 in the model in which the node does appear).

A document vector space consists of [17]:

- a collection of *documents* (two process models in our case);
- a set of *index terms* according to which the documents are indexed; and
- an *index vector* for each document, assigning a weight to each index term.

This leaves us to specify how *index terms* and *index vectors* are established in our case. We derive the *index terms* from the sets of activities, look-ahead links and look-back links of the causal footprints. However, where traditionally index terms are the same for all documents, they can differ for two causal footprints. In particular we use activities as index terms, but we want to consider that activity labels can differ while still representing the same activity. For example, the labels "enter client information" and "enter client's information" differ with respect to their labels, but could still be considered the same activity. Therefore, we use the match between the nodes from the two BPGs (as it can be computed using the metrics from the previous sections) as input for determining the index terms and index vectors. We then determine the set of index terms as follows.

Definition 16. Let B_1 and B_2 be two BPGs with causal footprints $G_1=(A_1,L_{lb,1},L_{la,1})$ and $G_2=(A_2,L_{lb,2},L_{la,2})$ and let $M: A_1 \nrightarrow A_2$ be a partial injective mapping that associates similar activities. We define the set of index terms as: $\Theta = M \cup (A_1 - \mathrm{dom}(M)) \cup \; L_{lb,1} \cup L_{la,1} \cup (A_2 - \mathrm{cod}(M)) \cup \; L_{lb,2} \cup L_{la,2}$. In the remainder we consider the sequence of index terms $\lambda_{|\Theta|}$.

For example, if we develop a causal footprint for the tasks from Fig. 3 the set of index terms contains among others (t_{11},t_{21}) from M, t_{12} from $(A_1 - \mathrm{dom}(M))$, $(\{t_{11}\},t_{12})$ from $L_{lb,1}$ and $(t_{11},\{t_{12}\})$ from $L_{la,1}$.

We determine the index vector for each BPG by assigning a weight to each index term. An index term can either be a mapped activity, an unmapped activity or a (look-ahead or look-back) link and we use different formulae to determine the weight for different types of terms. There are many possible ways in which the formulae can be defined. For example, we can simply assign a mapped activity the weight 1 and an unmapped activity the weight 0, but we can also assign a mapped activity a weight that represents the quality of the mapping. However, the approach to determine the best way of assigning the weights is to propose a formula for assigning weights and experimentally establish whether that formula performs better than the previous ones. After experimentation, we got the best results when assigning weights as follows. (More information about the experiments that we used can be found in Section 6.)

- We assign an unmapped activity the weight 0.
- We assign a mapped activity a weight that represents the similarity with the activity to which it is mapped, using one of the similarity functions from Section 2.3.
- We assign a link with a weight that exponentially decreases with the number of nodes in the link, using the rationale that links with fewer nodes are more informative than links with more nodes.

Using these principles, we define the index vectors of the BPGs as follows.

Definition 17. Let B_1 and B_2 be two BPGs with causal footprints $G_1=(A_1,L_{lb,1},L_{la,1})$ and $G_2=(A_2,L_{lb,2},L_{la,2})$, let $M: A_1 \nrightarrow A_2$ be a partial injective mapping that associates similar activities, let $\lambda_{|\Theta|}$ be a sequence of index terms as defined in Definition 16 and let Sim be one of the formulae from Section 2.3 that determines the node similarity of two mapped activity nodes. We define the index vectors, $\overrightarrow{g_1} = (g_{1,1},g_{1,2},\ldots,g_{1,|\Theta|})$ and $\overrightarrow{g_2} = (g_{2,1},g_{2,2},\ldots,g_{2,|\Theta|})$ for the two BPGs, such that for

each index term λ_j, for $1 \leq j \leq |\Theta|$ and for each $i \in \{1,2\}$ holds that:

$$g_{i,j} = \begin{cases} \mathrm{Sim}(a,a') & \text{if } \exists(a,a') \in M \\ & \text{such that } \lambda_j = a \vee \lambda_j = a' \\ \dfrac{\mathrm{Sim}(a,a')}{2^{|as|}} & \text{if } \exists(as,a) \in L_{lb,i} \\ & \text{such that } \lambda_j = (as,a) \\ & \text{and } (\exists(a,a') \in M \vee \exists(a',a) \in M) \\ \dfrac{\mathrm{Sim}(a,a')}{2^{|as|}} & \text{if } \exists(a,as) \in L_{la,i} \\ & \text{such that } \lambda_j = (a,as) \\ & \text{and } (\exists(a,a') \in M \vee \exists(a',a) \in M) \\ 0 & \text{otherwise} \end{cases}$$

For example, if we use syntactic label similarity to compute similarity of node pairs, then the index vector for the top BPG from Fig. 3 assigns $\mathrm{Simsyn}(t_{11},t_{21}) \approx 0.58$ to index term (t_{11},t_{21}) and $\mathrm{Simsyn}(t_{11},t_{21})/2^1 \approx 0.29$ to index term $(t_{11},\{t_{12}\})$.

Finally we can compute the behavioral similarity of the two BPGs, based on their causal footprints, using the cosine of the angle between their index vectors (which is a commonly accepted means for computing the similarity of two vectors [17]) as follows.

Definition 18. Let B_1 and B_2 be two BPGs with index vectors $\overrightarrow{g_1}$ and $\overrightarrow{g_1}$ as defined in Definition 17. We define their causal footprint similarity, denoted $\mathrm{simcf}(B_1,B_2)$, as:

$$\mathrm{simcf}(B_1,B_2) = \frac{\overrightarrow{g_1} \times \overrightarrow{g_2}}{|\overrightarrow{g_1}| \cdot |\overrightarrow{g_2}|}$$

Causal footprints do not capture the exact behavior of a process model but rather an approximation. If we used an exact representation of the process behavior—as captured by a Labelled Transition System (LTS) or a set of traces—we would run into computational complexity issues. Computing the LTS of a process model is exponential on the size of the model, and comparing two LTS for equivalence (using weak or branching bisimulation) is also exponential [18]. A similar remark applies if we use traces, with the additional issue that the set of traces of a process model with cycles is infinite. On the other hand, the computation of causal footprints is exponential (on the number of gateways), but the comparison between causal footprints can be done in linear time since it involves a simple aggregation of two vectors. The calculation of the footprints can be done incrementally when the business process models are added to the repository, and afterwards, the search itself can be done in $O(N \times M)$ where N is the number of models and M is the size of the largest model.

6. Empirical evaluation

In this section, we present an evaluation of the proposed similarity metrics in the context of *similarity search*. In our context, the similarity search problem is defined as follows: Given a process model P (the *query model*) and a collection of process models C (the *document models*), retrieve the models in C that are most similar to P and rank them according to their degree of similarity. There are at least two scenarios where similarity search is relevant:

1. Model repository maintenance: Before adding a model to a repository, a process analyst may wish to check that a similar model does not already exist, so as to prevent duplication and to take advantage of reuse opportunities. Similarly, a process analyst may wish to find out if a repository contains overlapping models. Such overlaps are often introduced because process analysts "copy/paste" existing model fragments when designing new process models.
2. Model alignment and merging: In the context of company mergers, process analysts need to find overlapping processes across the merged companies in order identify opportunities for consolidation. Also, when deploying a new Enterprise System into an organization, the existing process models of the organization need to be compared with the "reference" process models supported by the Enterprise System in order to identify overlaps.

In the first scenario, the query model and the document models are generally designed by the same team of process analysts and using the naming conventions and vocabulary of a given organization. As a result, the models are expected to be homogeneous, meaning that the task labels used in the query models and in the repository models are likely to be drawn from the same set. In the second scenario, the query model and the document models are designed by entirely independent teams. As a result, the task labels are heterogeneous. Below, we evaluate the proposed similarity metrics in each of these scenarios.

6.1. Evaluation with homogeneous labels

For this evaluation, we used the SAP reference model: a collection of 604 business process models (described as EPCs) that represent the business processes supported by the SAP ERP system. From this repository, we randomly extracted 100 business process models and tagged them as the "document models". We then randomly extracted 10 models from these 100 models. These models became the query models after undergoing the modifications described below. The reason for modifying the query models was to study the effect of different types of variations in labeling, structure and behaviour on the precision and recall of the proposed metrics.

- Query models 1 and 2 were left unchanged.
- Query models 3 and 4 were modified by changing the labels of the functions and the events of the original models into different labels that, to a person, mean the same (e.g. change "evaluated receipt settlement" into "carry out invoicing arrangement"). This variation is intended to serve as a challenge to label matching.
- Query models 5 and 6 were modified by taking a subgraph of the original model. For both models the

subgraph was approximately half the size of the original. This represents a structural variation while behaviour is unchanged.
- Query models 7 and 8 were modified by changing the connectors of the original model into connectors of different types. Each connector was changed to a random type that was different from the original type. This is a behavioural variation with minimal structural impact.
- Query models 9 and 10 were modified by re-ordering the functions and events in the model. Re-ordering was done by random swaps of two functions or of two events. This modification changes both structure and behaviour.

We performed the experiments by applying each of the metrics defined in this paper to perform a similarity search for each of the 10 query models. In other words, each of the query models was compared with each of the 100 document models and the results were ranked from highest to lowest similarity score.

We then manually determined the relevance for each of the 1000 possible ("query model", "document model") pairs. To this end, we rated the similarity of each of the 1000 pairs on a 1–7 Likert scale. Pairs that received a score of 5 or higher ("somewhat similar" to "very similar") were considered relevant, meaning that a query containing the "query model" should return the "document model" in question. Since we did this rating ourselves, we had to establish that there was no bias in our relevance judgments. To this end, we extracted a subset of 50 pairs and presented them to 20 process modeling experts, asking them to rate the similarity of each pair on the same Likert scale. We subsequently calculated the inter-rater dependency between our own judgement and those of the 20 experts using the Pearson correlation coefficient. The correlation was very strong (0.95 Pearson correlation coefficient) with 99% confidence, showing that our judgment was consistent with that of other process modeling experts.

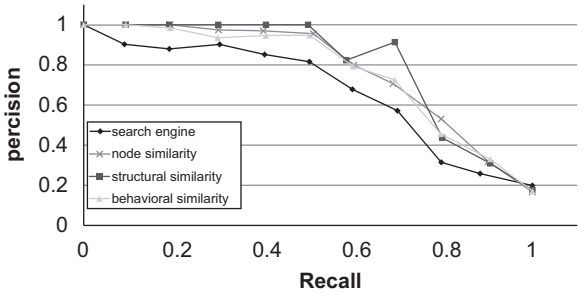

Fig. 9. Precision–recall curve (precisions are averaged across all 10 queries).

At first glance, one could think that because the query models and the document models are derived from the same collection, the results are predictable: Each query model should have a high similarity with exactly one document model and a low similarity with all other document models. However, this is far from being the case because the SAP reference model contains many overlapping processes. For example, there are seven variations of the procurement process. Overall, out of the 1000 (search model, document model) pairs, 108 pairs were judged as "similar" or "very similar" during the manual inspection.

As a baseline for comparison, we used a text-based search engine (namely the Indri search engine [19]). For each search and for each document model, we derived a file containing the list of function and event labels appearing in that model. We then loaded the document models into the search engine, and submitted each of the query models to obtain a ranked list of document models.

Fig. 9 and Table 1 summarize the results of the experiments. Fig. 9 shows the average precision and recall scores across all the queries in the recall intervals $[0 \ldots 0.05\rangle, [0.05 \ldots 0.15\rangle, [0.15 \ldots 0.25\rangle, \ldots, [0.95 \ldots 1.0\rangle$. Table 1 shows a more concise representation of the overall performance. For each metric it lists the mean average precision, this is the mean of the average precision of each query model. The average precision for a given query is the average of the precision scores obtained after each relevant document model is found [20]. The table also lists the first 10 precision and the first 20 precision, which are the precision for the first 10 or 20 search results, respectively. The graph and the table show that on average, the metrics from this paper perform better than a text-based search engine, thereby showing the use of such metrics.

Table 1 also shows the execution times observed when running all 10 queries on the repository comprising all 100 document models. The tests were conducted on a laptop with a dual core Intel processor, 2.53 GHz, 3 GB memory, running Microsoft Vista and SUN Java Virtual Machine version 1.6. In order to factor away one-off setup times, we ran the queries twice in a row and measured only the second run. The results show that the proposed techniques have sub-second execution times, which is acceptable considering that overall 1000 comparisons need to be performed. The execution times of the search engine are greater than node matching and structural similarity. Closer examination revealed that this is due to

Table 1
Overall results of experiments.

	Mean average precision	First 10 precision	First 20 precision	Exec. time
Search engine	0.76	0.70	0.46	826 ms
Node matching	0.80	0.79	0.44	109 ms
Structural similarity	0.83	0.78	0.48	208 ms
Behavioral similarity	0.80	0.74	0.46	40 s

the fact that the search engine request requires a call through the Java Native Interface, which implies dynamic code loading of native code. The execution of the behavioral similarity metrics is significantly higher than the other techniques, but as discussed in Section 5 this is because of the time required to construct the causal footprint of each document model. This computation can be performed incrementally as the document models are inserted into a repository.

Fig. 10 shows the average precision for each of the query models and each of the metrics. The graph shows that the metrics defined in this paper outperform text-based search engines when: (i) the query model is a subgraph of the document models it is meant to match (query models 5 and 6); or (ii) the query model and the document models it is meant to match, only differ in the types of connectors employed (query models 7 and 8).

Looking more closely at the results gives an indication as to why the metrics in this paper perform better when the query model is a subgraph of a document model. Text-based search algorithms rank a document (model) higher when a term from the search (model) appears more often. However, the metrics in this paper rank a document model higher when a term (or rather a function or event) from the document model appears in a frequency that is closer to the frequency with which it appears in the query model. For example, for query model 1 the document model that was identical to the query model was only the fifth hit in the text-based search algorithm, while it was the first hit both in the structural similarity metric and in the behavioral similarity metric. This effect is stronger in subgraph query models. For example, suppose that a query model is about billing clients and that it only has a single function "bill client". Also, suppose that there are two document models: one that is about negotiating a contract agreement, which contains the functions "negotiate with client", "send draft to client" and "send final offer to client"; and one that is about billing a client for received goods, which contains the functions "ship goods" and "bill client". A text-based search algorithm would rank the first document model higher, because the terms from the query model appear more frequently in that document model. The search metrics from this paper would rank the second document model higher, because there is a better match between functions in that document model.

We now consider the effects of varying the parameters of the proposed node similarity metrics. The node matching similarity metrics are parameterized by a threshold. Two nodes can be matched only if their similarity is above a given threshold. Furthermore, the similarity of two nodes can be determined using syntactic, semantic, contextual or attribute similarity as explained in Section 2.3. We tested the performance of these node similarity metrics for different thresholds in the context of a structural similarity metric. The results of these tests are shown in Fig. 11. This graph plots the *mean average precision* of different variants of node matching. The horizontal axis corresponds to different values of the threshold. Three curves correspond to syntactic, semantic and contextual similarity. The fourth curve corresponds to a similarity metric in which syntactic similarity is counted for a factor 0.75 and contextual similarity for a factor 0.25. The results for attribute similarity are not shown. We could not test the performance of the attribute similarity metrics, because our dataset did not contain any attributes. Testing the performance of this metric is left for future work.

The graph shows that contextual similarity performs significantly worse than the other forms of similarity. Consequently, contextual similarity can be used to improve the performance of the other forms of similarity—as illustrated by the curve that displays the combined syntactic/contextual similarity metric—but it is not useful by itself. The graph also shows of the use of semantic similarity improves the mean average precision of the technique only slightly.

We acknowledge that these results are dependent on the type of process models being compared: In process repositories where the event and function labels are standardized—e.g. based on the process classification framework of the American Productivity and Quality

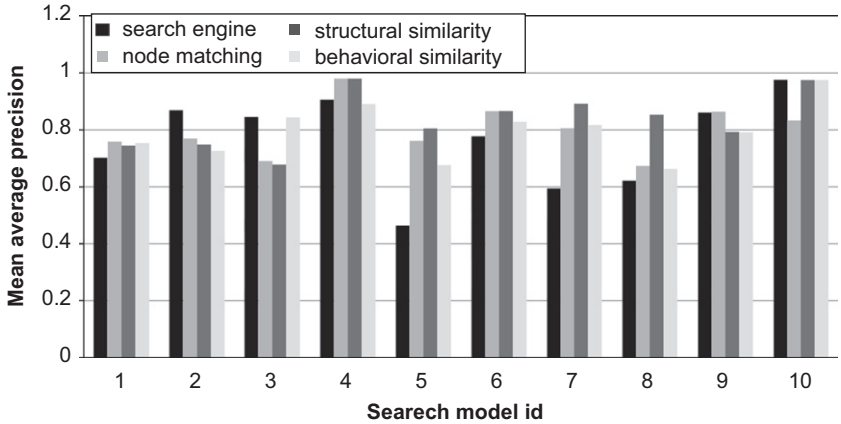

Fig. 10. Average precision per query model.

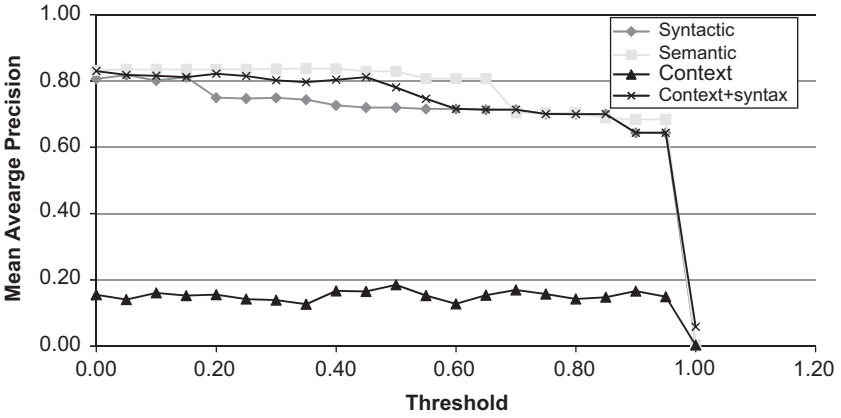

Fig. 11. Mean avg. precision of label matching variants.

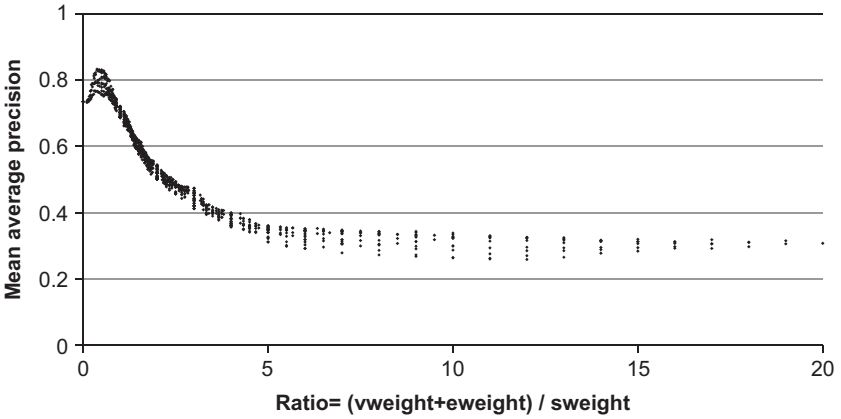

Fig. 12. Mean avg. precision of structural similarity variants.

Center[2]—the use of approximate label matching might be less crucial than in scenarios where significant variations in terminology exist. We used these results to parameterize the process similarity metrics. The average precision of the label matching technique shown in Figs. 9 and 10 are those obtained using syntactic similarity only and using a threshold of 0.5. We chose to use syntactic similarity with this setting, because it works very fast. The algorithm for computing syntactic similarity is much faster than that for semantic similarity because it does not have to do dictionary lookups. In addition to that using a threshold of 0.5 excludes many potential node matches, while not significantly degrading performance.

Structural similarity has three parameters: the weight given to edge deletion or insertion (eweight), the weight given to node deletions or insertion (vweight), and the weight given to substitution of a node. We tested all combinations of values of these three parameters between 0 and 1 in steps of 0.1—i.e. (0,0,0), (0, 0, 0.1), (0, 0, 0.2), …, (0, 0.1, 0), (0, 0.2, 0), etc. For each combination, we measured the mean average precision across the 10 search queries. After analyzing the results, we discovered a correlation between the parameter values: the best results are obtained when the ratio (*vweight*+*eweight*)/*sweight* is between 0.2 and 0.8, with an optimum occurring when this ratio is between 0.4 and 0.5. In other words, the best settings are those where substitutions are given twice the weight of insertions and deletions. This trend is shown in the scatter plot in Fig. 12. Each point in the scatter plot represents one combination of parameter values. The *y*-coordinate of a point is given by the mean average precision obtained for the combination of values in question, while the *x*-coordinate is given by the ratio (*vweight*+*eweight*)/*sweight*.[3] Clearly, the peak can be seen on the left part of the graph between 0 and 1. The recall and mean average precision results for the structural similarity metrics previously shown in Figs. 9 and 10 are those obtained with *vweight*=0.1, *sweight*=0.8 and *eweight*=0.2.

6.2. Evaluation in heterogeneous labels

For the second evaluation, the query models were taken from the process model repository of a large Dutch manufacturing company. These models were related to

[2] http://www.apqc.org/

[3] Combinations for which *sweight* is zero are not shown since the denominator of the ratio is then zero.

procurement, logistics and order management processes. Although we had a larger pool of query models available, we randomly extracted 10 query models to keep the number of manual comparisons feasible. We then randomly extracted 100 document models from the procurement, logistics and order management branches of the SAP reference model. We restricted ourselves to these branches because they are aligned with the application domain of the query models. The manual comparison between the query models and the document models was performed by a team of students.

The query models were designed by a team of analysts without any knowledge of the SAP reference model and using the naming conventions and vocabulary of the company, which were different from those of the SAP reference model. To assess the heterogeneity between the task labels in the query models and those in the document models, we compared every task label in the query models with every task in the document models. Among all pairs (query model task label, document model task label) only 5% had a semantic similarity score of greater than 0.5 in the heterogeneous dataset, as opposed to 16% in the homogeneous dataset.

Fig. 13 and Table 2 summarize the results of the experiments for the heterogeneous case. Fig. 13 shows the precision–recall curve while Table 2 shows the aggregate results. The results reported for the structural similarity correspond to the best setting of *vweight*, *eweight* and *sweight*, and using syntactic and semantic similarity combined. Other settings gave slightly less mean average precision, but still in the range 0.58–0.6 with only some extreme settings giving lower accuracy. The results reported for node matching correspond to the setting with a threshold of 0.5. The results obtained for node matching were almost identical whether we used semantic and syntactic similarity combined, or syntactic similarity alone.

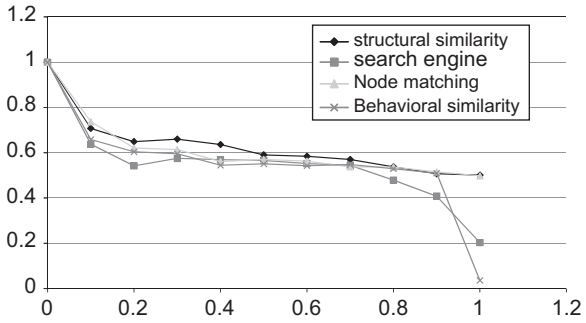

Fig. 13. Precision–recall curve for heterogeneous models.

The accuracy of all techniques is significantly lower than for the first dataset. This can be explained by the strong differences between the labels in the query model and those in the document models. Notwithstanding this heterogeneity, the similarity metrics described in this paper have better accuracy than the text-based search engine. The node matching and structural similarity techniques showed the best results—mean average precision of 0.6 for both techniques. The fact that node matching and structural similarity give almost the same results suggests that the topology of the graph is not as important as the number of matches between nodes in the query model and nodes in the document models. It appears that behavioral similarity performs less well on this dataset, suggesting again that taking into account the topology of the model (and thus the induced causal relations) does not add accuracy in the context of a heterogeneous dataset.

Table 2 also shows the execution times for computing the different similarity metrics, which are largely consistent with the performance results obtained for the homogeneous dataset.

Fig. 14 shows the average precision for each of the query models and each of the metrics for the heterogeneous dataset. The graph shows that the metrics defined in this paper outperform text-based search engines except in two cases. A close examination of the two models where the text-based search engine performed slightly better revealed that this models were smaller and simpler than the other query models. For larger models, label and structural similarity clearly outperform text-based search.

Overall, we can make the following observations:

- In the case where the query models and document models have labels extracted from the same space, the topology of the models and their induced behavior adds accuracy to the search technique.
- In the case where the query models and the document models are heterogeneous, the number of matches between nodes adds accuracy but not the topology or the induced behavior.
- In all cases, the proposed similarity metrics outperform a text-based search engine, suggesting that process model repositories need specialized indexing and search techniques.

7. Related work

In this section we discuss related work in three areas: (i) behavior similarity measurement (and in particular of behavior described in process models); (ii) information

Table 2
Aggregate results for heterogeneous models.

	Mean average precision	First 10 precision	First 20 precision	Exec. time
Search engine	0.53	0.55	0.54	610 ms
Node matching	0.6	0.64	0.58	265 ms
Structural similarity	0.6	0.65	0.61	283 ms
Behavioral similarity	0.56	0.56	0.55	6 min

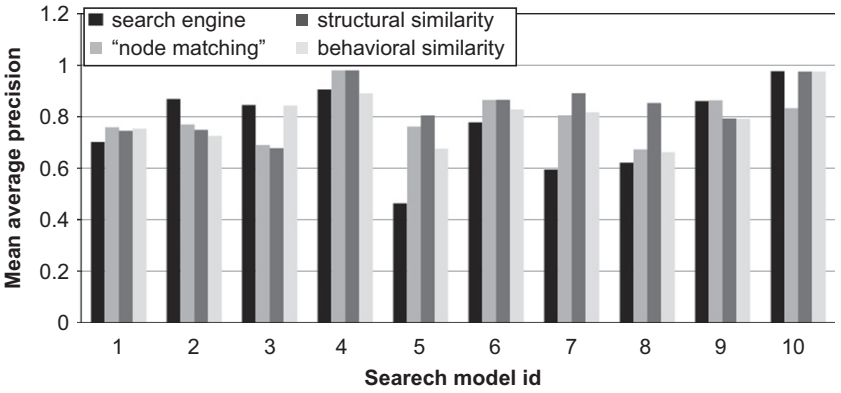

Fig. 14. Average precision per query model for heterogeneous models.

retrieval in process model repositories; and (iii) schema matching.

Existing work on determining a degree of similarity between process models is rather scarce, even though related topics like process model integration are well researched (see [21–25]). Also closely related are different notions of behavioral equivalence such as trace equivalence and bisimulation. While trace equivalence is based on a simple comparison of the sets of completed execution traces, bisimulation notions are stricter since they consider the points in time when decisions are taken, e.g., weak bisimulation equivalence is stricter than trace equivalence. A thorough overview of behavioral equivalence notions is presented in [26]. The strict "true or false" nature of these comparison techniques has been criticized in [27] as not appropriate for various application scenarios in the field of business process management. In addition, existing techniques for behavioral equivalence checking are mainly based on state-space analysis, which is computationally expensive: even the restricted class of 1-safe Petri nets require exponential space for most equivalence notions [28]. In our work, we avoid state space calculation by using causal footprints [29] as the basis for comparison. Since causal footprints capture constraints instead of the state space, this approach relates to constraint-based or property-based approaches to process modeling and verification [30–32]. Comparable concepts are used in [33] for service matching, an area where behavioral abstractions including interfaces [34], conversation protocols [7], behavioral signatures [35], and operating guidelines [36] are used.

The particular contribution of our paper is that it presents *and* validates a collection of similarity metrics. Most proposed business process similarity metrics either remain unvalidated, and do not take into account label similarity (by assuming that equivalent tasks have equivalent labels) nor behavioral similarity—focusing instead on structural similarity. Some work with all these feature has, though, been conducted for state charts and finite state automata. Nejati et al. [37] propose a similarity metric for computing the similarity of statecharts. It takes differences between labels of states into account and considers behavioral similarity, using approximations of bi-similarity as well as the nested structure of states in a statechart. Because of this latter feature, their technique is specific to statecharts. Their technique is validated using three pairs of statechart specifications of telecommunication services. The focus of their evaluation is not to determine the precision/recall of the technique as a search technique, but rather as a technique for merging pairs of statecharts. Wombacher [38] also empirically validates a number of metrics for measuring the similarity of workflows. His work focuses on workflows modeled using Finite State Automata and uses a validation by comparison with human judgement. Unfortunately, the approaches studied by Wombacher cannot be directly used for process models. Even though reachability graphs (which are basically automata) can be derived from process models, these can potentially be infinite or at least exponential in size of the process model [11]. Also, the evaluation conducted by Wombacher is not based on measures of precision or recall, but rather on the ability of different methods to reproduce the judgement of human experts based on 23 queries with a small number of possible solutions per query.

Other previous work deals with the comparison of business process models that are captured in notations similar to EPCs or BPMN. Li et al. [39] propose a structural approach to determine the similarity between business process models. Their approach first determines "atomic" differences between business process models and subsequently groups these differences using patterns that they have defined in earlier work [40]. They then compute a similarity value based on the grouped differences. Their structural algorithm, therefore, clearly differs from ours. However, in their paper, they do not provide a validation of their algorithm, so it is not possible to compare their performance to ours. In a similar vein, Minor et al. [41] use graph–edit distance to find an appropriate *changed* business process model in case a running business process model needs to be changed. As a result it does not consider differences between task labels, while our algorithm takes it into account. Lu and Sadiq [42] introduce an algorithm for measuring similarity of process variants, based on process model fragments. It is targeted towards querying collections of process model variants for variants with certain features. These features can cover other aspects than tasks and their relations, such as use of

Table 3
Overview of related work.

	Comparison between	Label Sim.	Struct. Sim.	Behav. Sim.	Validation
This paper	Processes	√	√	√	√
Nejati et al. [37]	State charts	√	√	√	√
Wombacher [38]	Finite state machines	√	√	√	√
Li et al. [39]	Processes	x	√	x	x
Minor et al. [41]	Processes	x	√	x	x
Lu and Sadiq [42]	Processes	x	√	x	x
Van der Aalst et al. [27]	Log and process	√	x	√	x
Ehrig et al. [13]	Processes	√	√	x	x
Madhusudan et al. [49]	Processes	√	√	x	x

resources and timing characteristics. In their work, van der Aalst et al. [27] calculate a degree of behavioral similarity for measuring the fitness of a set of event logs relative to a process model. This degree helps to optimize the match between model and log in the area of process mining (see [43,44]). Finally, Ehrig et al. [13] match task labels based on structural and semantic properties, among other options using WordNet synonyms [45]. While this technique is close to our semantic matching technique, it has not been validated, but applied as a tool for recommendation support [46].

The problem of process model similarity search can be related to that of schema matching [47]. There are, however, important differences between process models and schemas. Firstly, data models and schemas generally have labelled edges (associations or schema elements) in addition to labelled nodes. Secondly, the types of nodes and the attributes attached to nodes are different in process models when compared to schemas or data models (e.g. there are no control nodes in data models). During our experiments, we implemented a graph matching technique originally designed for schema matching, namely similarity flooding [48]. After adapting the technique to deal with process models, we tested it on the dataset discussed in this paper using various parameter settings. The similarity flooding technique led to a poor score—0.56 of mean average precision for the best settings (with a first-10 precision of 0.6). We attribute this poor performance to the fact that edges in process models lack labels, while schema matching techniques, such as similarity flooding, heavily rely on edge labels. Madhusudan et al. [49] introduce a structural metric for process model comparison based on similarity flooding. However, Madhusudan et al. rely on a semantic notation in which process models have labels attached to their edges. Also, Madhusudan et al. did not validate their technique experimentally.

Table 3 summarizes features of related research in comparison to the work reported in this paper. As can be seen, our paper provides a consolidation of different approaches to similarity calculation and a respective validation.

Different models have been developed to exploit structure in text for information retrieval (see [50]) and some of these concepts have been applied to process models. One example is the process query language (PQL) proposed in [51] which uses a structured description of a process in terms of OWL classes and attributes. Yet, in contrast to our work, the behavioral aspect of a process is not considered. The same holds for other query approaches such as business process query language (BPQL) by [52] or BPMN-Q by [53]. The work on the definition of query languages is complementary to our work. Our work focuses on optimal techniques for finding a good match based for a given query, where the query is simply a process model or a part thereof. The work on query languages focuses on defining a good language for specifying the query.

8. Conclusion

In this paper we presented three parameterized similarity metrics between business process models:

1. node matching similarity metrics that measure similarity based on properties of business process model elements, such as their labels and their other attributes;
2. structural similarity metrics that measure similarity based on the properties of business process model elements, as well as the relations between these elements; and
3. behavior similarity metrics that measure similarity based on the intended behavior of process models.

The implementation of the proposed metrics can be found in the "similarity plugin" of the ProM process mining and analysis framework.[4]

We experimentally evaluated the metrics by determining their precision and recall. We compared the performance of these metrics with that of a text-based search engine to establish a baseline of what process similarity metrics should be able to achieve in order to be useful. Our expectation was, when taking process structure and behavior into account, that process similarity metrics should out-perform text-based search engines.

We tested our expectation, by applying the different metrics for similarity search to two datasets. Both datasets consisted of 100 "document" business process models and 10 "search query" business process models. In both cases we used the 10 search queries to search the collection of documents, by returning the documents in order of their similarity to the search queries. Subsequently we used information retrieval metrics such as precision and recall to evaluate the performance of the different similarity search

[4] Available at: http://prom.sourceforge.net.

metrics. However, in the first dataset the search queries were obtained by taking 10 business process models from the collection and modifying those models in predefined manner (e.g.: by taking a sub-graph, or by replacing words in the labels by synonyms) to study the effect of certain properties of search models on the performance of each metric. In the second dataset the search queries were taken from a separate collection of models. Consequently, the two datasets illustrate two separate use cases of similarity search techniques: one in which models are compared to models from the same collection (e.g. searching for overlapping models) and one in which models are compared to models from another collection (e.g. searching for a reference model that matches an organization's own model).

Both experiments showed that all three metrics that we defined indeed outperform a text-based search engine for process model similarity queries. This was expected as the proposed metrics use knowledge of the structure and intended behavior of business process models. In case search query models are taken from the same collection as the document models, the structural similarity metric slightly outperforms the others. In addition, there is evidence that the metrics from this paper in particular perform better if the "search query" is a subgraph of one (or more) of the business process models in the collection, if the "search query" only differs from the document models it should match in the types of gateways employed and if the "search query" is relatively complex in terms of the number of its elements.

In this paper, we focused on developing similarity metrics for comparing pairs of process models, rather than efficient algorithms for similarity search. Still, the experimental evaluation shows that we can perform 1000 comparisons (10 query models times 100 document models) in sub-second times, except for the behavioral similarity metric. These results are encouraging, but in order to scale up to repositories with several hundred or thousands of models, more scalable techniques would be needed. A direction for future work is to design indexing structures for scaling up the proposed similarity metrics to larger datasets.

Another direction for future work is to use the metrics that are defined above to determine optimal mappings between the tasks of two similar business processes. This requires that we allow mapping between sets of tasks, rather than single tasks, because a set of tasks in one process may match with a set of tasks in the other process. Research showed that this is common when the two processes have been developed independent of each other [54].

Acknowledgments

The research that led to this paper is partly funded by the Beta Research School for Operations Management and Logistics at TU Eindhoven, and by the Estonian Centre of Excellence in Computer Science.

References

[1] M. Rosemann, Potential pitfalls of process modeling: part a, Business Process Management Journal 12 (2) (2006) 249–254.
[2] P. Hart, N. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Transactions on Systems Science and Cybernetics 4 (2) (1968) 100–107.
[3] B. Dongen, J. Mendling, W. Aalst, Structural patterns for soundness of business process models, in: Proceedings of the 10th IEEE EDOC Conference (EDOC), IEEE, 2006, pp. 116–128.
[4] B.F. van Dongen, R.M. Dijkman, J. Mendling, Measuring similarity between business process models, Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE), Lecture Notes in Computer Science, vol. 5074, Springer, 2008, pp. 450–464.
[5] R.M. Dijkman, M. Dumas, L. García-Bañuelos, Graph matching algorithms for business process model similarity search, in: U. Dayal, J. Eder, J. Koehler, H.A. Reijers (Eds.), BPM, Lecture Notes in Computer Science, vol. 5701, Springer, 2009, pp. 48–63.
[6] R.M. Dijkman, M. Dumas, L. García-Bañuelos, R. Kaarik, Aligning business process models, in: EDOC, IEEE Computer Society, Springer, 2009, pp. 45–53.
[7] D. Grigori, J. Corrales, M. Bouzeghoub, Behavioral matchmaking for service retrieval: application to conversation protocols, Information Systems 33 (7–8) (2008) 681–698.
[8] W. Aalst, A. Hofstede, B. Kiepuszewski, A. Barros, Workflow patterns, Distributed and Parallel Databases 14 (1) (2003) 5–51.
[9] N. Lohmann, E. Verbeek, R.M. Dijkman, Petri net transformations for business processes—a survey, T. Petri Nets and Other Models of Concurrency 2 (2009) 46–63.
[10] M. Fauvet, M.L. Rosa, M. Sadegh, A. Alshareef, R. Dijkman, L. García-Bañuelos, H. Reijers, W. van der Aalst, M. Dumas, J. Mendling, Managing process model collections with AProMoRe, in: ICSOC 2010 Demo Track, accepted for publication.
[11] A. Valmari, The state explosion problem, in: W. Reisig, G. Rozenberg (Eds.), Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996, Lecture Notes in Computer Science, vol. 1491, Springer, 1998, pp. 429–528.
[12] R. Dijkman, M. Dumas, C. Ouyang, Semantics and analysis of business process models in bpmn, Information and Software Technology (IST) 50 (12) (2008) 1281–1294.
[13] M. Ehrig, A. Koschmider, A. Oberweis, Measuring similarity between semantic business process models, in: APCCM '07: Proceedings of the Fourth Asia–Pacific Conference on Conceptual Modelling, Australian Computer Society Inc., Darlinghurst, Australia, Australia, 2007, pp. 71–80.
[14] B. van Dongen, R. Dijkman, J. Mendling, Measuring similarity between business process models, BETA Working Paper WP-233, Eindhoven University of Technology, 2007.
[15] I. Levenshtein, Binary code capable of correcting deletions, insertions and reversals, Cybernetics and Control Theory 10 (8) (1966) 707–710.
[16] M.F. Porter, An algorithm for suffix stripping, Program 14 (3) (1980) 130–137.
[17] G. Salton, A. Wong, C. Yang, A vector space model for automatic indexing, Communications of the ACM 18 (11) (1975) 613–620.
[18] J.F. Groote, F.W. Vaandrager, An efficient algorithm for branching bisimulation and stuttering equivalence, in: 17th International Colloquium on Automata, Languages and Programming (ICALP), Springer, Warwick University, UK, 1990, pp. 626–638.
[19] D. Metzler, W. Croft, Combining the language model and inference network approaches to retrieval, Information Processing and Management 40 (5) (2004) 735–750.
[20] C. Buckley, E.M. Voorhees, Evaluating evaluation measure stability, in: Proceedings of the 23rd Annual International ACM SIGIR Conference, ACM, New York, NY, USA, 2000, pp. 33–40.
[21] G. Preuner, S. Conrad, M. Schrefl, View integration of behavior in object-oriented databases, Data & Knowledge Engineering 36 (2) (2001) 153–183.
[22] T. Basten, W. Aalst, Inheritance of behavior, Journal of Logic and Algebraic Programming 47 (2) (2001) 47–145.
[23] G. Grossmann, Y. Ren, M. Schrefl, M. Stumptner, Behavior based integration of composite business processes, in: W. van der Aalst, B. Benatallah, F. Casati, F. Curbera (Eds.), Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5–8, 2005, Proceedings, Lecture Notes in Computer Science, vol. 3649, Springer, 2005, pp. 186–204.
[24] V. Pankratius, W. Stucky, A formal foundation for workflow composition, workflow view definition, and workflow normalization based on petri nets, in: S. Hartmann, M. Stumptner (Eds.), Conceptual Modelling 2005 Second Asia–Pacific Conference on Conceptual Modelling (APCCM2005), Newcastle, NSW, Australia,

January/February 2005, CRPIT, vol. 43, Australian Computer Society, 2005.
[25] J. Mendling, C. Simon, Business process design by view integration, in: J. Eder, S. Dustdar (Eds.), Proceedings of BPM Workshops 2006, Lecture Notes in Computer Science, vol. 4103, Springer-Verlag, Vienna, Austria, 2006, pp. 55–64.
[26] R.J. van Glabbeek, U. Goltz, Refinement of actions and equivalence notions for concurrent systems, Acta Informatica 37 (4/5) (2001) 229–327.
[27] W. Aalst, A. Medeiros, A. Weijters, Process equivalence: comparing two process models based on observed behavior, in: S. Dustdar, J. Fiadeiro, A. Sheth (Eds.), Proceedings of BPM 2006, Lecture Notes in Computer Science, vol. 4102, Springer-Verlag, Vienna, Austria, 2006, pp. 129–144.
[28] J. Esparza, Decidability and complexity of Petri net problems—an introduction, in: W. Reisig, G. Rozenberg (Eds.), Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996, Lecture Notes in Computer Science, vol. 1491, Springer, 1998, pp. 374–428.
[29] B. Dongen, J. Mendling, W. Aalst, Structural patterns for soundness of business process models, in: Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference E (DOC'06), IEEE, Hong Kong, China, 2006, pp. 116–128.
[30] Z. Manna, A. Pnueli, The Temporal Logic of Reactive and Concurrent Systems: Specification, Springer-Verlag, New York, 1991.
[31] H. Eertink, W. Janssen, P. Oude Luttighuis, W. Teeuw C. Vissers, A business process design language, in: J. Wing, J. Woodcock, J. Davies (Eds.), World Congress on Formal Methods, Lecture Notes in Computer Science, vol. 1708, Springer, 1999, pp. 76–95.
[32] M. Pesic, M. Schonenberg, N. Sidorova, W. van der Aalst, Constraint-based workflow models: change made easy, in: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, OTM Confederated International Conferences CoopIS, DOA, ODBASE, GADA, and IS 2007, Vilamoura, Portugal, November 25–30, 2007, Proceedings, Part I, Lecture Notes in Computer Science, vol. 4803, Springer, 2007, pp. 77–94.
[33] R. Eshuis, P.W.P.J. Grefen, Structural matching of BPEL processes, in: ECOWS, IEEE Computer Society, 2007, pp. 171–180.
[34] L. Kuang, J. Wu, Y. Li, S. Deng, Z. Wu, Exploring dependency between interfaces in service matchmaking, in: IEEE SCC, IEEE Computer Society, 2007, pp. 506–513.
[35] Z. Shen, J. Su, Web service discovery based on behavior signatures, in: IEEE SCC, IEEE Computer Society, 2005, pp. 279–286.
[36] C. Stahl, K. Wolf, Deciding service composition and substitutability using extended operating guidelines, Data & Knowledge Engineering 68 (9) (2009) 819–833.
[37] S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, P. Zave, Matching and merging of statecharts specifications, in: 29th International Conference on Software Engineering (ICSE 2007), ACM/IEEE, 2007, pp. 54–63.
[38] A. Wombacher, Evaluation of technical measures for workflow similarity based on a pilot study, Proceedings of the 14th International Conference on Cooperative Information Systems (CoopIS'06), Lecture Notes in Computer Science, vol. 4275, Springer, Berlin, 2006, pp. 255–272.
[39] C. Li, M.U. Reichert, A. Wombacher, On measuring process model similarity based on high-level change operations, Technical Report TR-CTIT-07-89, Centre for Telematics and Information Technology, University of Twente, Enschede, December 2007.
[40] B. Weber, M. Reichert, S. Rinderle-Ma, Change patterns and change support features—enhancing flexibility in process-aware information systems, Data & Knowledge Engineering 66 (3) (2008) 438–466.
[41] M. Minor, A. Tartakovski, R. Bergmann, Representation and structure-based similarity assessment for agile workflows, in: R. Weber, M. Richter (Eds.), 7th International Conference on Case-Based Reasoning, Lecture Notes in Artificial Intelligence, vol. 4626, Springer, Heidelberg, Germany, 2007, pp. 224–238.
[42] R. Lu, S. Sadiq, On the discovery of preferred work practice through business process variants, Conceptual Modeling—ER 2007, Lecture Notes in Computer Science, vol. 4801, Springer, Heidelberg, Germany, 2007, pp. 165–180.
[43] G. Greco, A. Guzzo, G. Manco, D. Saccà, Mining and reasoning on workflows, IEEE Transactions on Knowledge and Data Engineering 17 (4) (2005) 519–534.
[44] W. Aalst, H. Reijers, A. Weijters, B. Dongen, A. Medeiros, M. Song, H. Verbeek, Business process mining: an industrial application, Information Systems 32 (5) (2007) 713–732.
[45] G. Miller, WordNet: a lexical database for English, Communications of the ACM 38 (11) (1995) 39–41.
[46] T. Hornung, A. Koschmider, G. Lausen, Recommendation based process modeling support: method and user experience, in: Q. Li, S. Spaccapietra, E.S.K. Yu, A. Olivé (Eds.), ER, Lecture Notes in Computer Science, vol. 5231, Springer, 2008, pp. 265–278.
[47] E. Rahm, P.A. Bernstein, A survey of approaches to automatic schema matching, VLDB Journal 10 (4) (2001) 334–350.
[48] S. Melnik, H. Garcia-Molina, E. Rahm, Similarity flooding: a versatile graph matching algorithm and its application to schema matching, in: International Conference on Data Engineering, IEEE Computer Society, Los Alamitos, CA, USA, 2002, pp. 117–128.
[49] T. Madhusudan, L. Zhao, B. Marshall, A case-based reasoning framework for workflow model management, Data & Knowledge Engineering 50 (1) (2004) 87–115.
[50] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, ACM Press/Addison-Wesley, 1999.
[51] M. Klein, A. Bernstein, Towards high-precision service retrieval, IEEE Internet Computing 8 (1) (2004) 30–36.
[52] M. Momotko, K. Subieta, Process query language: a way to make workflow processes more flexible, in: G. Gottlob, A. Benczúr, J. Demetrovics (Eds.), Advances in Databases and Information Systems, 8th East European Conference, ADBIS 2004, Budapest, Hungary, September 22–25, 2004, Processing, Lecture Notes in Computer Science, vol. 3255, Springer, 2004, pp. 306–321.
[53] A. Awad, G. Decker, M. Weske, Efficient compliance checking using BPMN-Q and temporal logic, in: M. Dumas, M. Reichert, M.-C. Shan (Eds.), Proceedings of the 6th International Conference on Business Process Management, Lecture Notes in Computer Science, vol. 5240, Springer, 2008, pp. 326–341.
[54] M. Weidlich, R. Dijkman, J. Mendling, The ICoP framework: identification of correspondences between process models, in: Proceedings of the 22th International Conference on Advanced Information Systems Engineering (CAiSE), Springer, 2010, pp. 483–498.