Similarity of Business Process Models: Metrics and Evaluation

Remco Dijkman, Marlon Dumas, Boudewijn van Dongen, Reina Käärik, Jan Mendling

Abstract—It is common for large and complex 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 process models in the repository that most closely resemble a given process model or fragment thereof. The paper presents three similarity metrics that can be used to answer such queries: (i) label matching similarity that compares the labels 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 similarity metrics are experimentally evaluated in terms of precision and recall, and in terms of correlation of the metrics with respect to human judgement. The experimental results show that all three metrics yield comparable results, with structural similarity slightly outperforming the other two metrics. Also, all three metrics outperform traditional search engines when it comes to searching through a repository for similar business process models.

Index Terms—I.5.3 [Pattern Recognition]: Clustering - Similarity measures. H.3.3 [Information Storage And Retrieval]: Information Search and Retrieval Retrieval models,Relevance feedback.

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 and use 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 search engines can be used to index and to search business process model repositories. However, traditional search engines are based on keyword search and text similarity. They are clearly useful in situations where a user is looking for a model that contains an activity with a certain keyword in its label. On the other hand, it is unclear how far search engines are also appropriate for process model similarity queries, since they do not take into account the structure and behavioral semantics of process models.

This paper studies three similarity metrics designed to answer process model similarity queries. The first metric is a label-based one. It exploits the fact that process models are composed of labeled nodes. The metric starts 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 metric is structural. It uses existing techniques for graph comparison based on graph-edit distance [2]. This metric takes into account both the node labels and the topology of the process models. The third metric is behavioral. It takes into account the behavioral semantics of process models, specifically, the causal relations between activities in a process model. These causal relations are represented in the form of a *causal footprint* [3].

The paper is an extended and revised version of our earlier work [4] in which we introduced the behavioral

- *R. Dijkman is with the School of Industrial Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.*
  E-mail: r.m.dijkman@tue.nl
- *M. Dumas and R. Käärik are with the Institute of Computer Science, University of Tartu, J Liivi 2, 50409 Tartu, Estonia.*
  E-mail: marlon.dumas@ut.ee, reinak@ut.ee
- *B. van Dongen is with the Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.*
  E-mail: b.f.v.dongen@tue.nl
- *J. Mendling is with the Institute of Information Systems, Humboldt-University Berlin, Spandauer Straße 1, 10178 Berlin, Germany.*
  E-mail: contact@mendling.com

similarity notion and we evaluated it using a dataset consisting of 50 pairs of models. In this paper, we propose two other notions of process model similarity and we present a more extensive evaluation using a traditional search engine as a baseline for comparison. The evaluation is done in two ways: firstly using the classical notions of precision and recall, and secondly by calculating statistical correlations between the similarity scores given by each metric, and those given by human experts. The evaluation results show that similarity metrics that take into account the structure and behavior of process models outperform search engines at answering process model similarity queries.

The remainder of the paper is structured as follows. Section 2 presents the notation used to represent business process models. Section 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 Event-driven Process Chain (EPC) notation [5] for business process modeling. 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 Process Modeling and EPCs

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. In this paper, we use EPCs as a process modeling notation. Firstly, EPCs are representative of other process modeling notations insofar as they include the most common constructs found in other notations. Secondly, EPCs have been used in industrial practice for almost two decades. As a result, many large process model repositories are available as EPCs. In particular, the repository we use in our experiments is composed of EPCs. Still, we do not compromise on generality: the label similarity metrics defined in this paper is independent from the specific process modeling notation used, while the structural similarity metrics can be applied to any graph-based process modeling notation and the behavioral similarity metrics works for any notation that can be mapped to causal footprints. There exist mappings from EPCs and from Workflow nets to causal footprints, and mappings exist from BPMN and from BPEL to Workflow nets.

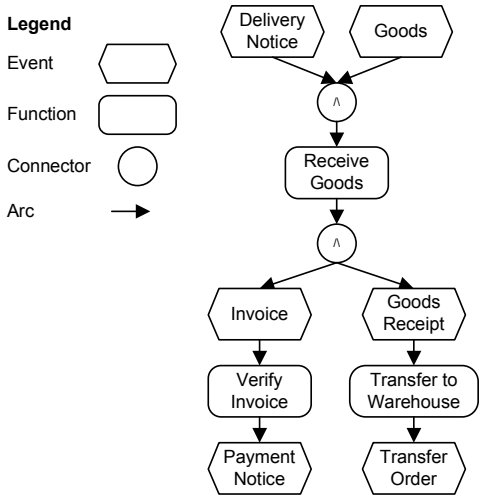

Fig. 1. Example EPC

The EPC notation is a graph-based language for documenting the temporal and logical dependencies between functions and events in an organization. An EPC consist of functions, events and connectors that can be connected by arcs. Functions represent activities that can be performed in an organization. A function has exactly one incoming and one outgoing arc. Events represent the pre-conditions (what must have occurred before a function can be performed) and post-conditions (what has occurred after a function has been performed) of functions. An event can have no incoming arcs and one outgoing arc, in which case we call it a *start event*, one incoming arc and no outgoing arcs, in which case we call it an *end event*, or one incoming and one outgoing arc. Connectors represent logical decision points. Connectors can either have one incoming arc and multiple outgoing arcs, in which case we call it a *split*, or multiple incoming arcs and one outgoing arcs, in which case we call it a *join*. There are three types of connectors: AND (denoted $\wedge$), OR (denoted $\vee$) and XOR (denoted $\times$). Figure 1 shows an example of a business process in the EPC notation describing how received goods are transferred to the warehouse and the respective invoice is verified. We define an EPC as follows.

Definition 1 (EPC). *An EPC is a tuple* (F, E, C, l, A) *and* Ω *a set of text labels, in which:*

- F *is a finite set of functions;*
- E *is a finite set of events;*
- C *is a finite set of connectors;*
- $l : (F \cup E \to \Omega) \cup (C \to \{\text{and}, \text{xor}, \text{or}\})$ *labels functions and events with text and connectors with types;*
- $A \subseteq (F \cup E \cup C) \times (F \cup E \cup C)$ *is the set of arcs.*

An EPC is syntactically correct if and only if it contains at least one function and has strict alternation of events and functions on each path of arcs from start to end.

Informally, the behavior of an EPC can be described as follows. Start events can occur from the start of the process. Other events and functions can occur when their incoming arc is enabled. When they occur their outgoing arc (if any) is enabled. Connectors can be used to create more advanced behavior. An AND-split enables all outgoing arcs when its incoming arc is enabled. Similarly, an XOR-split enables one outgoing arc and an OR-split enables one or more outgoing arcs. An AND-join waits for all its incoming arcs to be enabled, before enabling its outgoing arc. Similarly, an XOR-join waits for one of its incoming arcs and an OR-join waits for all incoming arcs that can be enabled (which is non-trivial from a formal point of view [6], [7]).

Using these semantics, we can describe the behavior of the example EPC in Figure 1. Initially, this EPC waits for a 'delivery notice' and 'goods'. When both these events have occurred, the subsequent AND-join enables its outgoing arc and therewith the 'receive goods' function can occur. (We also say that the function's pre-condition in terms of the events that precede it is satisfied.) After the function has been performed, an AND-split causes both the 'invoice' and 'goods receipt' events to occur. (We also say that the post-condition of the function, namely that an 'invoice' and a 'goods receipt' notice is produced, is satisfied.) The EPC continues in this way until it completes.

In the remainder of this paper we will use the notions of path and connector chain to discuss the relations between events and functions.

Definition 2 (Paths and Connector Chains)**.** *Let (F, E, C, l, A) be an EPC and $N = F \cup E \cup C$ be the set of nodes of that EPC. For each node $n \in N$, we define* path *$a \hookrightarrow b$ refers to the existence of a sequence of EPC 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: $(n_1, n_2), (n_2, n_3), \ldots, (n_{k-1}, n_k) \in A$. This includes the empty path of length zero, i.e., for any node $a : a \hookrightarrow a$. Let $M \subseteq N$ be a set of nodes and $a \neq b \in N$ be two nodes. A path containing only $n_2, \ldots, n_{k-1} \in M$, denoted $a \overset{M}{\hookrightarrow} b$ is called a restricted path. This includes the empty restricted path, i.e., $a \overset{M}{\hookrightarrow} b$ if $(a, b) \in A$. The path restricted to the set of connectors, denoted $a \overset{C}{\hookrightarrow} b$, is called a* connector chain.

2.2 Causal Footprint of an EPC

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 [8]. 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 representation of behavior.

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 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 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 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 an EPC if and only if it is consistent with the behavior of that EPC.

Definition 4 (Causal Footprint of an EPC)**.** *Let $P = (F, E, C, l, A)$ be an EPC, $G = (F, L_{lb}, L_{la})$ be a causality graph over the functions of P, and $W \subseteq F^*$ be the set of possible orders in which functions from P can be performed. G is a causal footprint of PC 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 $0 \leq i \leq n-1$ with $\sigma[i] = a$, there is a $j : i < j \leq n - 1$, 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 $0 \leq i \leq n - 1$ with $\sigma[i] = b$, there is a $j : 0 \leq j < i$, such that $\sigma[j] \in A$,*

For example, a possible causal footprint for the EPC from figure 1 has 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 an EPC, because there are multiple EPCs that have the same causal footprint (for example, the EPC that can be derived from figure 1 by transforming the AND-split in an XOR-split). Also, there are multiple possible causal footprints for this EPC.

We refer to [3] for an algorithm to compute a causal footprint of an EPC.

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

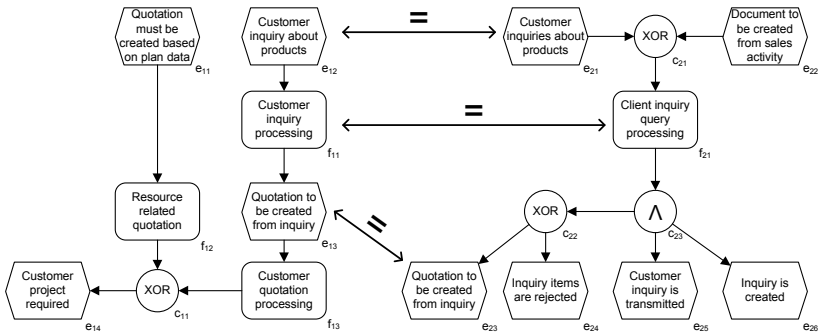

Fig. 2. *Customer Inquiry* and *Customer Inquiry and Quotation Processing* EPCs.

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. Figure 2 is an example in point: functions "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 three ways of measuring similarity between elements of different process models:

1) Syntactic similarity, where we consider only the syntax of the labels,
2) Semantic similarity, where we abstract from the syntax and look at the semantics of the words within the labels, and
3) Contextual similarity, where we do not only consider the labels of the elements themselves, but the context in which these elements 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, Koschmider and Oberweis [9] 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. A comparison is presented in [10].

2.3.1 Syntactic Similarity

Given two labels (e.g. the labels of functions and/or events in an EPC), the syntactic similarity metrics returns the degree of similarity as measured by the string-edit distance. The string-edit distance [11] is the number of atomic string operations necessary to get from one string to another. These atomic string operation include: removing a character, inserting a character or substituting a character for another.

Definition 5 (Syntactic similarity)**.** *Let* $(E_1, F_1, C_1, l_1, A_1)$ *and* $(E_2, F_2, C_2, l_2, A_2)$ *be two disjoint EPCs. Furthermore let* $n_1 \in F_1 \cup E_1 \cup C_1$ *and* $n_2 \in F_2 \cup E_2 \cup C_2$ *be two nodes from those EPCs and let* $l_1(n_1)$ *and* $l_2(n_2)$ *be the two strings that represent the labels of those nodes, i.e. we can calculate their length, denoted* $|l_1(n_1)|$ *and* $|l_2(n_2)|$*, and their edit distance, denoted* $ed(l_1(n_1), l_2(n_2))$*. We define the* syntactic similarity *of EPC nodes* n_1 *and* n_2 *as follows:*

$$syn(n_1, n_2) = 1 - \frac{ed(l_1(n_1), l_2(n_2))}{\max(|l_1(n_1)|, |l_2(n_2)|)}$$

For example, the syntactic similarity between the events e_{12} and e_{21} from figure 2 with labels "Customer inquiry about product" and "Customer inquiries about product" from figure 1 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 function 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. We assume an exact match is preferred over a match on synonyms. Accordingly words that are identical are given an equivalence score of 1, while words that are synonymous are given an equivalence score of 0.75 (see justification below). Hence, the semantic similarity score is defined as follows.

Definition 6 (Semantic similarity)**.** *Let* $(E_1, F_1, C_1, l_1, A_1)$ *and* $(E_2, F_2, C_2, l_2, A_2)$ *be two disjoint EPCs. Furthermore let* $n_1 \in F_1 \cup E_1 \cup C_1$ *and* $n_2 \in F_2 \cup E_2 \cup C_2$ *be two nodes from those EPCs and let* $w_1 = l_1(n_1)$ *and* $w_2 = l_2(n_2)$ *be the labels of those nodes (and assume that* w_1 *and* w_2 *are sets of words, i.e. we denote the number of words by* $|w_1|$ *and* $|w_2|$ *and we can use standard set operators). We define the* semantic similarity *of EPC nodes* n_1 *and* n_2 *as follows:*

$$sem(n_1, n_2) = \frac{1.0 \cdot |w_1 \cap w_2| + 0.75 \cdot \sum_{\substack{s \in w_1 \setminus w_2 \\ t \in w_2 \setminus w_1}} synonym(s, t)}{\max(|w_1|, |w_2|)}$$

Where $synonym$ *is a function that returns* 1 *if the given words are synonyms and* 0 *if they are not.*

For example, consider the functions f_{11} and f_{21} from figure 2 with labels "Customer inquiry processing" and "Client inquiry query processing". These labels which consist of the collections of words w_1 =["Customer","inquiry","processing"] and w_2 =["Client", "inquiry", "query", "processing"], respectively. We only need to consider a synonym mapping between $w_1 \setminus w_2$ = ["Customer"] and $w_2 \setminus w_1$ = ["Client","query"]. We consider "Customer" and "Client" synonymous and "Customer" and "query" not synonymous. Therefore, the semantic similarity between w_1 and w_2 equals
$sem(w_1, w_2) = \frac{1.0 \cdot 2 + 0.75 \cdot (1+0)}{4} \approx 0.69$.

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 [12]. Stemming reduces words to their stem form. For example, "stemming", "stemmed" and "stemmer" all become "stem".

We selected the 0.75 weight factor for synonyms experimentally. Specifically, we manually compared 210 function pairs taken from the SAP Reference Model and for each pair, we determined if their labels matched or not 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 Contextual Similarity

The two metrics defined above focus on the similarity of two process model elements. We now define a third 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 for EPCs, because in EPCs functions are always preceded and succeeded by events. Thus, when comparing two functions, the contextual similarity metric takes the surrounding events into account. Another process modeling technique for which contextual similarity is particularly useful is Petri nets, because in Petri nets 'transitions' are always preceded and succeeded by places (and vice versa).

We refer to preceding model elements as the *input context* and to succeeding model elements as the *output context* of another model element.

Definition 7 (Input and output context)**.** *Let (E, F, C, l, A) be an EPC. For a node $n \in F \cup E$, we define the input context $n^{in} = \{n' \in F \cup E \mid n' \overset{C}{\hookrightarrow} n\}$ and the output context $n^{out} = \{n' \in F \cup E \mid n \overset{C}{\hookrightarrow} n'\}$*

To determine the contextual similarity between elements of a business process model, we need a mapping between the elements in their input and output contexts. Such a mapping is in itself based on a similarity metric, for example one of the metrics from section 2.3.1 or 2.3.2, and is called an *equivalence mapping* as defined below.

Definition 8 (Equivalence Mapping)**.** *Let L_1, L_2 be two disjoint sets. Furthermore, let $\mathrm{s} : L_1 \times L_2 \to [0..1]$ be a similarity function such that for all $l_1 \in L_1$ and $l_2 \in L_2$: $\mathrm{s}(l_1, l_2) = \mathrm{s}(l_2, l_1)$. A partial injective mapping $M_\mathrm{s} : L_1 \nrightarrow L_2$ is an* equivalence mapping*, if and only if for all $l_1 \in L_1$ and $l_2 \in L_2$: $M(l_1) = l_2$ implies that $\mathrm{s}(l_1, l_2) > 0$.*

An optimal equivalence mapping *$M_\mathrm{s}^{opt} : L_1 \nrightarrow L_2$ is an equivalence mapping, such that for all other equivalence mappings M holds that*
$\sum_{(l_1,l_2) \in M_\mathrm{s}^{opt}} \mathrm{s}(l_1, l_2) \geq \sum_{(l_1,l_2) \in M_\mathrm{s}} \mathrm{s}(l_1, l_2)$.

For example, in figure 2 we can develop an equivalence mapping between $\{e_{12}\}$ 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 $syn(e_{12}, e_{22}) \approx 0.24$. $M_{syn}^{opt} = \{(e_{12}, e_{21})\}$ is the optimal equivalence mapping, because $syn(e_{12}, e_{21}) = 0.90$. The only other possible mapping is the empty mapping.

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

Definition 9 (Contextual Similarity)**.** *Let $(E_1, F_1, C_1, l_1, A_1)$ and $(E_2, F_2, C_2, l_2, A_2)$ be two disjoint EPCs. Let $n_1 \in F_1$ and $n_2 \in F_2$ be two functions and let* Sim *be one of the similarity functions from section 2.3.1 or 2.3.2. Furthermore, let $M_{\mathrm{Sim}}^{optin} : n_1^{in} \nrightarrow n_2^{in}$ and $M_{\mathrm{Sim}}^{optout} : n_1^{out} \nrightarrow b_2^{out}$ be two optimal equivalence mappings between the input and output contexts of n_1 and n_2 respectively. We define the contextual similarity as follows:*

$$con(n_1, n_2) = \frac{|M_{\mathrm{Sim}}^{optin}|}{2 \cdot \sqrt{|n_1^{in}|} \cdot \sqrt{|n_2^{in}|}} + \frac{|M_{\mathrm{Sim}}^{optout}|}{2 \cdot \sqrt{|n_1^{out}|} \cdot \sqrt{|n_2^{out}|}}$$

In the remainder of this paper, we use $\mathrm{Sim}(n_1, n_2)$ to denote the similarity value between two elements of a model. Any of the symmetric similarity functions above (syn, sem or con) can be substituted for this, as well as any (weighted) combination thereof, as long as the sum of weights is 1.

3 Label Matching Similarity

The first similarity measure we study, namely *label matching similarity*, is based on pairwise comparisons of node labels. It is obtained by calculating an optimal equivalence mapping between the nodes of the two process models being compared. The label 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 10 (Label Matching Similarity)**.** *Let $P_1 = (F_1, E_1, C_1, l_1, A_1)$ and $P_2 = (F_2, E_2, C_2, l_2, A_2)$ be two EPCs and let* Sim *be a function that assigns a similarity score*

to a pair of functions/events. Let $M^{opt}_{Sim} : (F_1 \nrightarrow F_2) \cup (E_1 \nrightarrow E_2)$ *be an optimal equivalence mapping derived from* Sim*. The* label matching similarity *between* P_1 *and* P_2 *is:*

$$simlbm(P_1, P_2) = \frac{2 \cdot \Sigma_{(n,m) \in M^{opt}_{Sim}} \mathrm{Sim}(n, m)}{|F_1| + |F_2| + |E_1| + |E_2|}$$

The label matching similarity metrics is parameterized by the similarity metrics used to compare pairs of labels. We can use the semantic or syntactic similarity notions defined in Section 2.3, or a weighted average of them. Note that we can not use the context similarity metrics, because this latter is only defined for pairs of functions, while here we need to compare functions and events.

We further parameterize the label 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 8, this means that instead of enforcing that $\mathrm{s}(l_1, l_2) > 0$, we enforce that $\mathrm{s}(l_1, l_2) \geq \mathrm{threshold}$.

As an example, consider the EPCs from Figure 2. The optimal equivalence mapping between these EPCs is denoted by the two-way arrows with the $=$ symbol on them. Assuming that we use syntactic equivalence (syn) to determine the similarity between the functions and events, and that we use a threshold of 0.5, the similarity score of the elements included in the equivalence mapping is: $syn(e_{12}, e_{21}) = 0.90$, $syn(f_{11}, f_{21}) \approx 0.58$ and $syn(e_{13}, e_{23}) = 1.00$. 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 label matching similarity between these two EPCs is:

$$\frac{2 \cdot \Sigma_{(n,m) \in M^{opt}_{syn}} syn(n, m)}{|F_1| + |F_2| + |E_1| + |E_2|} = \frac{2 \cdot (0.90 + 0.58 + 1.00)}{3 + 1 + 4 + 6} \approx 0.35$$

The problem of finding an optimal equivalence mapping can be reduced to the linear assignment problem [13] as follows. We define a new graph in which the set of nodes is the union of the set of functions in the two EPCs. In this graph, an edge exists between each function in the first EPC and each function in the second EPC. Each edge is assigned a weight equal to the similarity score between the labels of the functions that it connects. If the similarity between two functions is below the threshold, we assign a weight of zero to the edge linking these functions, so that even if they are matched, this pair of functions will not contribute to the overall similarity score. Coming back to the example in Figure 2, an edge is defined between f_{11} and each of the three functions in the first EPC. The edge (f_{11}, f_{21}) is given a weight of 0.58, while the edges (f_{12}, f_{21}) and (f_{13}, f_{21}) are given a weight of zero because the syntactic similarity scores between these pairs of functions are below the threshold (which we assume is 0.5).

By following this procedure, we obtain a complete weighted bipartite graph over the set of functions in the EPCs being compared. The problem of finding an optimal equivalence mapping is then reduced to that of finding a maximum-weight matching over this bipartite graph, so that every node in the smaller side of the bipartite graph is matched with exactly one node in the other side. This problem can be solved using the Hungarian method or Jonker & Volgenant's algorithm [13]. In our experiments, we used an implementation of Jonker & Volgenant's algorithm. In the working example, the algorithm will naturally match f_{11}, with f_{21} and no other function pairs.

We then apply the same procedure to find an optimal equivalence mapping between the events in one EPC and those in the second. The union of the two optimal equivalence mappings (the one over functions and the one over events) is then calculated, and the label matching similarity score is calculated as per Definition 10.

4 Structural Similarity

The second similarity metric we study is a similarity metric over the structure of EPCs, by considering an EPC as a labeled graph. If we consider an EPC as a graph, then its functions, events and connectors are nodes of the graph and the arcs are edges of the graph. In the case of functions and events, their labels become the labels of the corresponding nodes. In the case of connectors, the type of a connector (and, xor, or) becomes the label of the node corresponding to this connector. We can then assign a similarity score to two EPCs by computing their graph-edit distance [2]. 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 label matching similarity, graph-edit distance is obtained by first computing a mapping between the EPC nodes 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 11 (Graph Edit Distance)**.** *Let* $P_1 = (F_1, E_1, C_1, l_1, A_1)$ *and* $P_2 = (F_2, E_2, C_2, l_2, A_2)$ *be two EPCs. Let* $N_1 = F_1 \cup E_1 \cup C_1$ *be the nodes of* P_1 *and* $N_2 = F_2 \cup E_2 \cup C_2$ *be the nodes of* P_2 *and let* Sim

be one of the similarity metrics from subsection 2.3. Let $M : (F_1 \nrightarrow F_2) \cup (E_1 \nrightarrow E_2) \cup (C_1 \nrightarrow C_2)$ *be a partial injective mapping that maps functions, events and connectors.*

Let $n \in N_1 \cup N_2$ *be a node.* n *is substituted if and only if* $n \in \mathrm{dom}(M)$ *or* $n \in \mathrm{cod}(M)$. 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 $(n, m) \in A_1$ *be an edge.* (n, m) *is inserted in or deleted from* P_1 *if and only if there do not exist mappings* $(n, n') \in M$ *and* $(m, m') \in M$ *and edge* $(n', m') \in A_2$*. Edges that are inserted in or deleted from* P_2 *are defined similarly.* se *is the set of all inserted or deleted edges.*

The distance induced by the mapping is defined as:

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

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

As an example, consider the EPCs from figure 2. Assuming that we use syntactic equivalence (syn) to determine the similarity between functions and events, the distance of the mapping that is displayed in the figure is: $12+16+2\cdot(1-0.90+1-0.58+1-1.00) \approx 29,04$

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 12 (Graph Edit Distance Similarity)**.** *Let* $P_1 = (F_1, E_1, C_1, l_1, A_1)$ *and* $P_2 = (F_2, E_2, C_2, l_2, A_2)$ *be two EPCs. Let* $N_1 = F_1 \cup E_1 \cup C_1$ *be the nodes of* P_1 *and* $N_2 = F_2 \cup E_2 \cup C_2$ *be the nodes of* P_2 *and let* Sim *be one of the similarity metrics from subsection 2.3.*

Furthermore, let $M : (F_1 \nrightarrow F_2) \cup (E_1 \nrightarrow E_2) \cup (C_1 \nrightarrow C_2)$ *be a mapping that induces the graph edit distance between the two processes and let* sn *and* se *be defined as in definition 11. We define the* graph edit distance similarity *as:*

$$simged(P_1, P_2) = 1 - \overline{\{\mathrm{snv}, \mathrm{sev}, \mathrm{sbv}\}}$$

Where:

$$\mathrm{snv} = \frac{|\mathrm{sn}|}{|N_1|+|N_2|}$$
$$\mathrm{sev} = \frac{|\mathrm{se}|}{|A_1|+|A_2|}$$
$$\mathrm{sbv} = \frac{2\cdot\Sigma_{(n,m)\in M} 1-\mathrm{Sim}(n,m)}{|N_1|+|N_2|-|\mathrm{sn}|}$$

Instead of using a plain average of the three components, we can use a weighted average. Another possible variation is to ignore certain nodes of an EPC. We ignore nodes by removing them from the EPC and replacing paths through ignored nodes by direct arcs.

Definition 13 (Node abstraction)**.** *Let* $P = (F, E, C, l, A)$ *be an EPC, let* $N = F \cup E \cup C$ *be its nodes,* Ω *be the set of all possible labels and let* $I \subseteq N$ *be the subset of nodes to ignore. The EPC in which the nodes from* I *are ignored is the EPC* $P' = (F - I, E - I, C - I, l - (I \times \Omega), A')$*, where* $A' = \{(a, b) | a, b \in (N - I), a \overset{I}{\hookrightarrow} b, a \neq b\}$.

For example, when using graph edit distance similarity on figure 2, all arcs are inserted or deleted, leading to the maximal edit distance with respect to arcs. However, there are indirect arcs from e_{21} to f_{21} and from f_{21} to e_{23}. 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 connector nodes can lead to incorrect similarity measurements. This issue can be addressed by ignoring all connector nodes, but of course that would mean that connector nodes are not considered in the similarity metric at all.

Note that the result of ignoring nodes in an EPC does not necessarily lead to a well-formed EPC. For example, in the result there can be direct arcs from functions to functions and from events to events. Furthermore, this way of ignoring nodes can lead to functions and events with multiple incoming and outgoing arcs. All of this is not allowed in a well-formed EPC

5 Behavioral Similarity

The third similarity metric we study is a similarity metric over the behavior of EPCs. The benefit of using behavioral similarity over structural similarity is illustrated by the issue that is addressed at the end of section 4, namely that indirect relations via the inserted or deleted connectors are not considered in structural similarity, while they are relevant. In behavioral similarity indirect relations *are* considered. For example in the behavior of the EPCs from figure 2 there is a direct relation between event e_{21} and function f_{21} (i.e. e_{21} is in the look-back link of f_{21}), 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 EPCs, by computing their distance in the document vector space [14] that can be constructed from their causal footprints. We use a document vector space model of the causal footprints of the EPCs, rather than of the EPCs themselves, to incorporate an approximation of behavior in the similarity metric. We have to use an approximation or behavior rather than the actual behavior (as it can, for example, be defined in terms of a state transition system or a set of traces), because the computational complexity of computing the actual behavior of an EPC is NP-complete and leads to large, possibly infinite, results. The computation of causal footprints is also expensive, but the comparison between causal footprints can be done efficiently since it involves comparing two lists. Therefore, footprint calculation can be done when the EPC is added to that collection and when it is modified. This does not have to affect the search time of a search algorithm that is defined based on the similarity metric explained in this section.

A document vector space consists of [14]:

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

This leaves us to define how *index terms* and *index vectors* are established in our case.

We derive the *index terms* from the sets of functions, 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 EPCs. In particular we use EPC functions as index terms, but we want to consider that EPC function labels can differ while still representing the same function. For example, the function labels "enter client information" and "enter client's information" differ with respect to their labels, but could still be considered the same function. Therefore, we use the match between the functions from the two EPCs (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 14. *Let P_1 and P_2 be two EPCs with causal footprints $G_1 = (F_1, L_{lb,1}, L_{la,1})$ and $G_2 = (F_2, L_{lb,2}, L_{la,2})$ and let $M : F_1 \nrightarrow F_2$ be a partial injective mapping that associates similar functions. We define the set of index terms as: $\Theta = M \cup (F_1 - \mathrm{dom}(M)) \cup L_{lb,1} \cup L_{la,1} \cup (F_2 - \mathrm{cod}(M)) \cup L_{lb,2} \cup L_{la,2}$. In the remainder we consider the sequence of index terms $\lambda_{|\Theta|}$.*

For example, for figure 2 the set of index terms is $\{(f_{11}, f_{12}), f_{12}, f_{13}, (\{f_{11}\}, f_{13}), (f_{11}, \{f_{13}\})\}$.

We determine the index vector for each EPC by assigning a weight to each index term. An index term can either be a mapped function, an unmapped function 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 function the weight 1 and an unmapped function the weight 0, but we can also assign a mapped function 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 function the weight 0.
- We assign a mapped function a weight that represents the similarity with the function 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 EPCs as follows.

Definition 15. *Let P_1 and P_2 be two EPCs with causal footprints $G_1 = (F_1, L_{lb,1}, L_{la,1})$ and $G_2 = (F_2, L_{lb,2}, L_{la,2})$, let $M : F_1 \nrightarrow F_2$ be a partial injective mapping that associates similar functions, let $\lambda_{|\Theta|}$ be a sequence of index terms as defined in definition 14 and let* Sim *be one of the formulae from subsection 2.3 that determines the label similarity of two mapped functions. 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 EPCs, such that for each index term λ_j, for $1 \leq j \leq |\Theta|$ and for each $i \in \{1, 2\}$ holds that:*

$$g_{i,j} = \begin{cases} \mathrm{Sim}(f, f') & \text{if } \exists (f, f') \in M \\ & \text{such that } \lambda_j = f \vee \lambda_j = f' \\ \frac{\mathrm{Sim}(f,f')}{2^{|fs|-1}} & \text{if } \exists (fs, f) \in L_{lb,i} \\ & \text{such that } \lambda_j = (fs, f) \\ & \text{and } (\exists (f, f') \in M \vee \exists (f', f) \in M) \\ \frac{\mathrm{Sim}(f,f')}{2^{|fs|-1}} & \text{if } \exists (f, fs) \in L_{la,i} \\ & \text{such that } \lambda_j = (f, fs) \\ & \text{and } (\exists (f, f') \in M \vee \exists (f', f) \in M) \\ 0 & \text{otherwise} \end{cases}$$

For example, if we use semantic label similarity to compute similarity of node pairs, then the index vector for the rightmost EPC from figure 2 assigns $sem((f_{11}, f_{12})) \approx 0.69$ to index term (f_{11}, f_{12}) and 0 to the other index terms.

Finally we can compute the behavioral similarity of the two EPCs, 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 [14]) as follows.

Definition 16. *Let E_1 and E_2 be two EPCs with index vectors $\overrightarrow{g_1}$ and $\overrightarrow{g_1}$ as defined in definition 15. We define their causal footprint similarity, denoted simcf(E_1, E_2), as:*

$$\mathit{simcf}(E_1, E_2) = \frac{\overrightarrow{g_1} \times \overrightarrow{g_2}}{|\overrightarrow{g_1}| \cdot |\overrightarrow{g_2}|}$$

6 Empirical Evaluation

There are different ways to evaluate the performance of the metrics that we discussed above. We use two forms of evaluation. Firstly, we consider traditional precision and recall measures that have been extensively used in information retrieval. In this way, we can relate the results to this body of knowledge. In this part of the evaluation, the precision and recall obtained using a search engine serves as a benchmark. Secondly, we are interested in the practical value of the similarity metrics. Accordingly, we investigate how close the metrics approximate human judgement of process model similarity. This is achieved by calculating the statistical correlation between the similarity scores yielded by each of the metrics, and a similarity score assigned by process modelers. Here, the human judgement is the benchmark.

6.1 Precision and Recall

To compute the precision and recall of our metrics, we applied them to the SAP reference model. This is a

collection of 604 business process models (described as EPCs) that represent the business processes supported by the SAP enterprise system. We randomly extracted 100 business process models from this collection and tagged them as the "document models" for our experiments. From the 100 documents we then randomly extracted 10 models. These models became the "search query models", after adapting them as follows.

- Search models 1 and 2 were left unchanged.
- Search models 3 and 4 were adapted 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'.)
- Search models 5 and 6 were adapted by taking a subgraph of the original model.
- Search models 7 and 8 were adapted by changing the connectors of the original model into connectors of different types.
- Search models 9 and 10 were adapted by re-ordering the functions and events in the model.

We made these adaptations to be able to check whether the properties above affect the precision and recall performance of the metrics.

We manually determined the relevance for each of the 1000 possible ("search query model", "document model") pairs. We then applied each of the metrics defined in this paper to perform a process model similarity query for each of the 10 search models. In other words, each of the search models was compared with each of the 1000 document models and the results were ranked from highest to lowest similarity score. The implementation of the proposed metrics can be found in the "similarity plugin" of the ProM process mining and analysis framework.[2]

We also used a traditional search engine (the Indri search engine [15]) to answer these same queries. 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 search models as a query, ranking the results from highest score to lowest score.

Figure 3 shows the average precision and recall scores across all the queries in the recall intervals $[0 \ldots 0.05\rangle, [0.05 \ldots 0.15\rangle, [0.15 \ldots 0.25\rangle, \ldots, [0.95 \ldots 1.0\rangle$. This graph shows that on average, the metrics from this paper perform better in terms of precision and recall for process model similarity queries.

Figure 4 shows the average precision for each of the search query models and each of the three metrics. The average precision for a given query is the average of the precision scores obtained after each relevant document model is found [16]. The graph show that the metrics defined in this paper outperform traditional search engines when: (i) the search query model is a subgraph of the document models it is meant to match (search models 5 and 6); or (ii) the search query model and the document models it is meant to match, only differ in the types of connectors employed (search models 7 and 8).

A concise representation of the overall performance of the metrics, expressed as the mean average precision over all 10 search queries, is listed in table 1.

Looking more closely at the results gives an indication as to why the metrics in this paper perform better when the search query model is a subgraph of a document model. Traditional search algorithms rank a document (model) higher when a term from the search query (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 search query model. For example, for search query model 1 the document model that was identical to the search query model was only the fifth hit in the traditional 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 search query models. For example, suppose that a search 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 traditional search algorithm would rank the first document model higher, because the terms from the search 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 similarity metrics. The label matching similarity metric is parameterized by a threshold. Two elements are matched only if the similarity of their labels is above the threshold. Furthermore, the similarity of two labels can be determined using syntactic similarity or semantic similarity as explained in Section 2.3. The average precision of the label matching technique shown in Figures 3 and Figure 4 are those obtained using syntactic similarity only and using a threshold of 0.5. We tested the label matching technique using other thresholds and using a combination of syntactic and semantic similarity (giving equal weight to the syntactic and the semantic similarity). The results of these tests are shown in Figure 5. This graph plots the *mean average precision* of different variants of the label matching technique. The horizontal axis corresponds to different values of the threshold. One curve corresponds plots the results using string-edit distance, while the second curve plots the results using both syntactic and semantic matching.

2. Available at: http://prom.sourceforge.net.

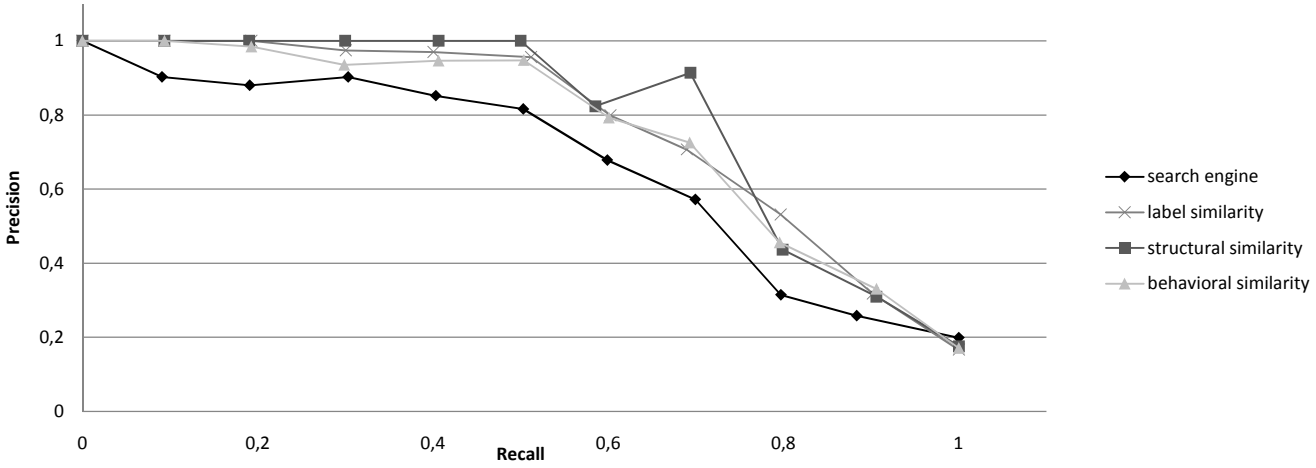

Fig. 3. Precision-recall curve (precisions are averaged across all 10 queries)

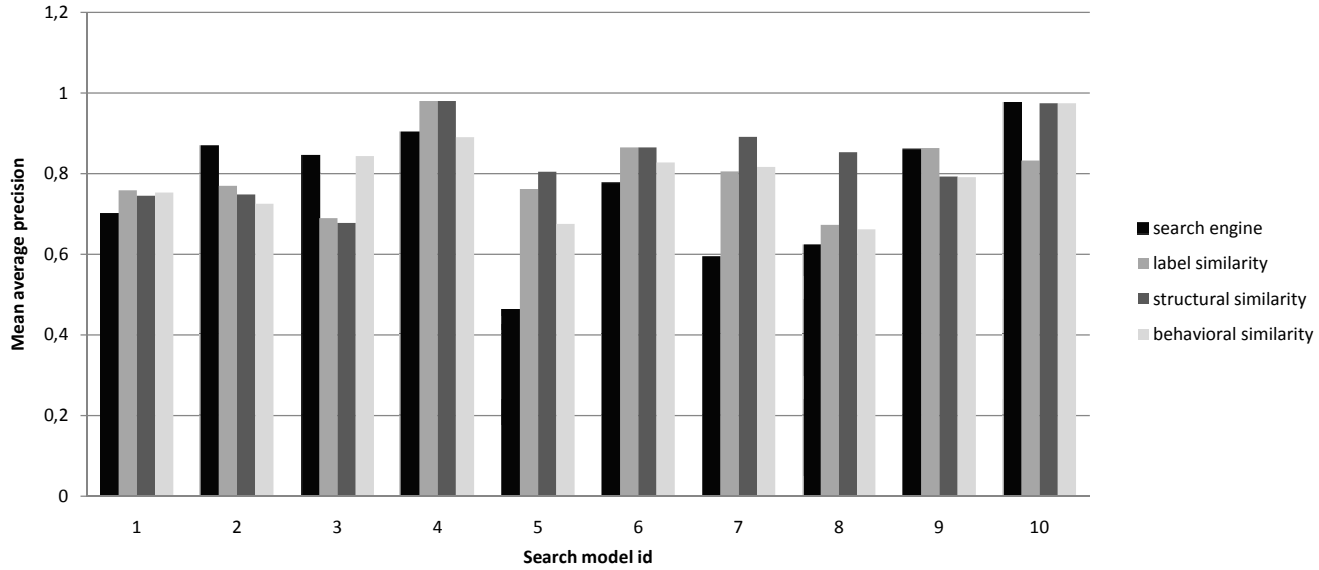

Fig. 4. Average precision per search query model

The graph shows that the mean average precision of the label matching technique varies little for thresholds below 0.75. For higher thresholds, the performance of the technique degrades rapidly. Indeed, past the 0.75 threshold, the technique will match two labels only if they are identical or almost identical, and most approximate matches are discarded. The graph also shows of the use of semantic similarity does not improve the precision of the technique, nor does it significantly degrade it. 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 Center[3] – the use of approximate label matching might be less crucial than in scenarios where significant variations in terminology exist.

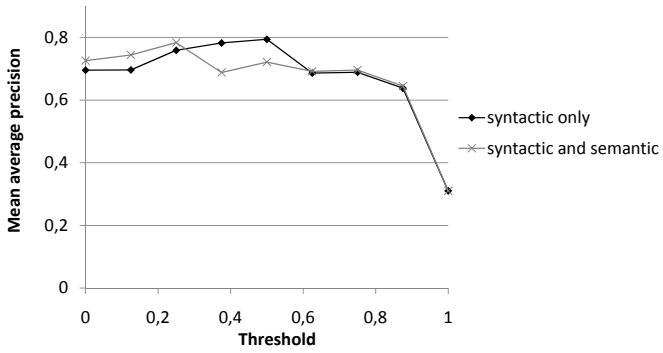

Fig. 5. Mean avg. precision of label matching variants

Graph edit distance similarity on the other hand 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 substitu-

3. http://www.apqc.org/

tion of a node. We tested all combinations of values of these three parameters between 0 and 1 in steps of 0.1 – i.e. $(0, 0, 0)$, $(0, 0, 1)$, $(0, 0, 0.2)$, $\ldots (0, 0.1, 0)$, $(0, 0.2, 0)$, 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 Figure 6. 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 $(\mathrm{vweight} + \mathrm{eweight})/\mathrm{sweight}$.[4] The recall and mean average precision results for the structural similarity metrics previously shown in Figures 3 and 4 are those obtained with $\mathrm{vweight} = 0.1, \mathrm{sweight} = 0.8 and \mathrm{eweight} = 0.2$.

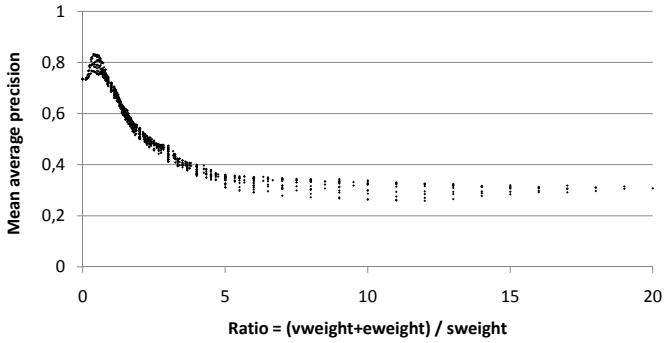

Fig. 6. Mean avg. precision of graph matching variants

6.2 Correlation with Human Judgment

The metrics from this paper can not only be applied to search process repositories, but also to directly measure the degree of similarity between a given pair of process models. The degree of similarity can be used to determine whether a match of a document model to a search query model is a viable match in the sense that the document model is worth considering at all (and not just the best match in a collection of document models). For example, if the metrics are used to search for a sales process that would be suitable to do business with, the result should not just be the "best" match in a collection, but also a viable match, in the sense that the process should be suitable to do business with.

To determine whether the metrics are fit to determine the viability of a match, we manually compared 1000 pairs of EPCs and assigned to each pair a similarity score on a 1 to 7 Likert scale.[5] We subsequently calculated the statistical correlation between the similarity scores produced by each of the three similarity metrics, and the human judgment (on a 1 to 7 Likert scale). The idea is that if there is a strong correlation between human judgment and the (similarity) scores produced by a metric, then the metric can be used to replace human judgment in order to make statements like: "these two process models are similar."

TABLE 1
Overall performance of the metrics

	Search Engine	Label Similarity	Structural Similarity	Behavioral Similarity
Mean Average Precision	0.76	0.80	0.83	0.80
Pearson Correlation Coefficient	0.03	0.72	0.72	0.73

Table 1 displays the correlations of the metrics with human judgment, using the Pearson correlation coefficient. All correlations were found to be significant. The table shows that there is a strong correlation between human judgment and both structural and behavioral similarity, but that there is virtually no correlation between human judgment and the search engine score. This can be explained, because a search engine score is not meant to represent "similarity", but rather to determine the best possible match (irrespective of whether that match is a viable match.)

7 Related Work

In this section we discuss related work in two areas. First, we focus on the usage of structured data to improve retrieval of process models. Second, we relate our contribution to previous work that addresses the problem of measuring similarity of behavior.

Different models have been developed to exploit structure in text for information retrieval (see [17]) and some of these concepts have been applied to process models. One example is the process query language (PQL) proposed in [18] 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 [19] or BPMN-Q by [20]. These query languages have in common that they return processes based on match/non-match. A ranking in terms of similarity is not considered.

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], [22], [23], [24], [25]). Also closely related are different notions of behavioral equivalence such as trace equivalence and bisimulation. While trace

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

5. These EPCs were the same ones that were used in the precision/recall experiments.

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], [31], [32].

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. [33] 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 [34] 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 [8]. 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, Reichert and Wombacher [35] 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 [36]. 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, Tartakovski and Bergmann [37] 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 [38] 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 resources and timing characteristics. Madhusudan, Zhao and Marshall [39] introduce a structural metric, which they call similarity flooding for workflow. It is based on the similarity flooding algorithm [40] that is introduced from the area of matching database schemas. The algorithm defines an initial mapping based on label similarity and then iteratively refines that mapping until a fixpoint is reached. It can deal with differences in task labels. However, it is not validated. Nor does it consider behavioral similarity (The original algorithm, however, is extensively validated.) In their work, Van der Aalst, Medeiros and Weijters [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 [41], [42]). Finally, there is work by Ehrig, Koschmider and Oberweis [43]. The authors match activity labels based on structural and semantic properties, among others using WordNet synonyms [44]. While this is close to our semantic match, it has not been validated.

Table 2 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.

8 Conclusion

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

1) label matching similarity metrics that measure similarity based on words in the labels of business process model elements;
2) structural similarity metric metrics that measure similarity based on the labels of business process model elements, as well as the relations between these elements; and

TABLE 2
Overview of Related Work

	Comparison between	Label Similarity	Structural Similarity	Behavioral Similarity	Validation
this paper	process models	√	√	√	√
Nejati et al. [33]	state charts	√	√	√	√
Wombacher [34]	finite state machines	√	√	√	√
Li et al. [35]	process models	x	√	x	x
Minor et al. [37]	process models	x	√	x	x
Lu and Sadiq [38]	process models	x	√	x	x
Madhusudan et al. [39]	process models	√	√	x	x
Van der Aalst et al. [27]	log and process model	√	x	√	x
Ehrig et al. [43]	process models	√	√	x	x

3) behavior similarity metrics that measure similarity based on the intended behavior of process models.

We experimentally evaluated these metrics by determining their precision and recall and their correlation with human judgement of similarity. We also compared the performance of these metrics with that of a traditional search engine with respect to the problem of process model similarity search.

We determined the precision and recall of the metrics by applying them to a collection of 100 business process models, using 10 business process models as 'search queries'. 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. These experiments showed that the structural similarity metric outperforms the others. Also, all three metrics that we defined outperform a traditional 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. 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 and if the 'search query' only differs from the document models it should match in the types of connectors employed.

We determined the correlation between the metrics and human judgement of similarity, by applying the metrics to a collection of 1000 pairs of business process models, for which we obtained human judgement of similarity beforehand. Correlation between the metrics and human judgement was strong and significant. There was no correlation between the relevance score obtained from the Indri search engine and human judgement.

Therefore, experimental evaluation lead to the conclusion that the similarity metrics defined in this paper have benefits over metrics from traditional search engines, both when using them to search a collection of business process models and when using them to determine the similarity between two business process models.

However, to be able to use the metrics to search a collection of business process models, work still has to be done. So far we have focused on developing the metrics, rather than efficient algorithms. The algorithms that we used to do the experimental evaluation linearly went through the collection of business process models to find the most relevant match. For 1000 comparisons (10 'search queries' times 100 models) this would take several minutes, which is not acceptable for practical search. Therefore, as future work we aim to develop efficient algorithms for searching collections of business processes for similar business processes.

Another direction for future work is to use the metrics that are defined above to determine optimal mappings between the activities of two similar business processes (the functions and events of two similar EPCs). This requires that we allow mapping between sets of activities, rather than single activities, because a set of activities in one process may match with a set of activities in the other process. Preliminary research even showed that this is common when the two processes have been developed independent of each other.

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*, vol. 12, no. 2, pp. 249–254, 2006.

[2] H. Bunke and K. Shearer, "A graph distance metric based on the maximal common subgraph," *Pattern Recognition Letters*, vol. 19, pp. 255–259, 1998.

[3] B. Dongen, J. Mendling, and 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, and J. Mendling, "Measuring similarity between business process models," in *Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE)*, ser. LNCS, vol. 5074. Springer, 2008, pp. 450–464.

[5] G. Keller, M. Nüttgens, and A. Scheer, "Semantische Process-modellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK)," Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrücken, 1992.

[6] E. Kindler, "On the semantics of EPCs: Resolving the vicious circle." *Data & Knowledge Engineering*, vol. 56, no. 1, pp. 23–40, 2006.

[7] J. Mendling, *Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness*, ser. Lecture Notes in Business Information Processing. Springer, 2008, vol. 6.

[8] A. Valmari, "The state explosion problem." in *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*, ser. Lecture Notes in Computer Science, W. Reisig and G. Rozenberg, Eds., vol. 1491. Springer, 1998, pp. 429–528.

[9] M. Ehrig, A. Koschmider, and A. Oberweis, "Measuring similarity between semantic business process models," in *APCCM '07: Proceedings of the fourth Asia-Pacific conference on Comceptual modelling*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2007, pp. 71–80.

[10] B. van Dongen, R. Dijkman, J. Mendling, and W. van der Aalst, "Detection of similarity between business process models," Eindhoven University of Technology, BETA Technical Report, 2007.

[11] I. Levenshtein, "Binary code capable of correcting deletions, insertions and reversals," *Cybernetics and Control Theory*, vol. 10, no. 8, pp. 707–710, 1966.

[12] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[13] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, pp. 325–340, 1987.

[14] G. Salton, A. Wong, and C. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[15] D. Metzler and W. Croft, "Combining the language model and inference network approaches to retrieval," *Information Processing and Management*, vol. 40, no. 5, pp. 735–750, 2004.

[16] C. Buckley and E. M. Voorhees, "Evaluating evaluation measure stability," in *Proceedings of the 23rd annual international ACM SIGIR conference*. New York, NY, USA: ACM, 2000, pp. 33–40.

[17] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[18] M. Klein and A. Bernstein, "Towards high-precision service retrieval," *IEEE Internet Computing*, vol. 8, no. 1, pp. 30–36, 2004.

[19] M. Momotko and K. Subieta, "Process query language: A way to make workflow processes more flexible." in *Advances in Databases and Information Systems, 8th East European Conference, ADBIS 2004, Budapest, Hungary, September 22-25, 2004, Proceesing*, ser. Lecture Notes in Computer Science, G. Gottlob, A. Benczúr, and J. Demetrovics, Eds., vol. 3255. Springer, 2004, pp. 306–321.

[20] A. Awad, G. Decker, and M. Weske, "Efficient compliance checking using bpmn-q and temporal logic," in *Proc. of the 6th International Conference on Business Process Management*, ser. Lecture Notes in Computer Science, M. Dumas, M. Reichert, and M.-C. Shan, Eds., vol. 5240. Springer, 2008, pp. 326–341.

[21] G. Preuner, S. Conrad, and M. Schrefl, "View integration of behavior in object-oriented databases." *Data & Knowledge Engineering*, vol. 36, no. 2, pp. 153–183, 2001.

[22] T. Basten and W. Aalst, "Inheritance of Behavior," *Journal of Logic and Algebraic Programming*, vol. 47, no. 2, pp. 47–145, 2001.

[23] G. Grossmann, Y. Ren, M. Schrefl, and M. Stumptner, "Behavior based integration of composite business processes." in *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings*, ser. Lecture Notes in Computer Science, W. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, Eds., vol. 3649. Springer, 2005, pp. 186–204.

[24] V. Pankratius and W. Stucky, "A formal foundation for workflow composition, workflow view definition, and workflow normalization based on petri nets," in *Conceptual Modelling 2005, Second Asia-Pacific Conference on Conceptual Modelling (APCCM2005), Newcastle, NSW, Australia, January/February 2005*, ser. CRPIT, S. Hartmann and M. Stumptner, Eds., vol. 43. Australian Computer Society, 2005.

[25] J. Mendling and C. Simon, "Business Process Design by View Integration," in *Proceedings of BPM Workshops 2006*, ser. Lecture Notes in Computer Science, J. Eder and S. Dustdar, Eds., vol. 4103. Vienna, Austria: Springer-Verlag, 2006, pp. 55–64.

[26] R. J. van Glabbeek and U. Goltz, "Refinement of actions and equivalence notions for concurrent systems," *Acta Inf.*, vol. 37, no. 4/5, pp. 229–327, 2001.

[27] W. Aalst, A. Medeiros, and A. Weijters, "Process Equivalence: Comparing two process models based on observed behavior," in *Proceedings of BPM 2006*, ser. Lecture Notes in Computer Science, S. Dustdar, J. Fiadeiro, , and A. Sheth, Eds., vol. 4102. Vienna, Austria: Springer-Verlag, 2006, pp. 129–144.

[28] J. Esparza, "Decidability and complexity of petri net problems - an introduction," in *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*, ser. Lecture Notes in Computer Science, W. Reisig and G. Rozenberg, Eds., vol. 1491. Springer, 1998, pp. 374–428.

[29] B. Dongen, J. Mendling, and W. Aalst, "Structural Patterns for Soundness of Business Process Models," in *Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06)*. Hong Kong, China: IEEE, 2006, pp. 116–128.

[30] Z. Manna and 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, and C. Vissers, "A business process design language," in *World Congress on Formal Methods*, ser. Lecture Notes in Computer Science, J. Wing, J. Woodcock, and J. Davies, Eds., vol. 1708. Springer, 1999, pp. 76–95.

[32] M. Pesic, M. Schonenberg, N. Sidorova, and W. van der Aalst, "Constraint-based workflow models: Change made easy," in *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*, ser. Lecture Notes in Computer Science, R. Meersman and Z. Tari, Eds., vol. 4803. Springer, 2007, pp. 77–94.

[33] S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, and P. Zave, "Matching and merging of statecharts specifications," in *29th International Conference on Software Engineering (ICSE 2007)*, 2007, pp. 54–63.

[34] A. Wombacher, "Evaluation of technical measures for workflow similarity based on a pilot study," in *Proc. 14th Int'l Conf. on Cooperative Information Systems (CoopIS'06)*, ser. LNCS, vol. 4275. Berlin: Springer, 2006, pp. 255–272.

[35] C. Li, M. U. Reichert, and A. Wombacher, "On measuring process model similarity based on high-level change operations," http://eprints.eemcs.utwente.nl/11574/, Centre for Telematics and Information Technology, University of Twente, Enschede, Technical Report TR-CTIT-07-89, December 2007.

[36] B. Weber, M. Reichert, and S. Rinderle-Ma, "Change patterns and change support features - enhancing flexibility in process-aware information systems," *Data Knowl. Eng.*, vol. 66, no. 3, pp. 438–466, 2008.

[37] M. Minor, A. Tartakovski, and R. Bergmann, "Representation and structure-based similarity assessment for agile workflows," in *7th International Conference on Case-Based Reasoning*, ser. LNAI, R. Weber and M. Richter, Eds., vol. 4626. Heidelberg, Germany: Springer, 2007, pp. 224–238.

[38] R. Lu and S. Sadiq, "On the discovery of preferred work practice through business process variants," in *Conceptual Modeling - ER 2007*, ser. LNCS, vol. 4801. Heidelberg, Germany: Springer, 2007, pp. 165–180.

[39] T. Madhusudan, L. Zhao, and B. Marshall, "A case-based reasoning framework for workflow model management," *Data Knowledge Engineering*, vol. 50, no. 1, pp. 87–115, 2004.

[40] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity flooding: A versatile graph matching algorithm and its application to schema matching," in *International Conference on Data Engineering*. Los Alamitos, CA, USA: IEEE Computer Society, 2002, pp. 117–128.

[41] G. Greco, A. Guzzo, G. Manco, and D. Saccà, "Mining and reasoning on workflows," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 4, pp. 519–534, 2005.

[42] W. Aalst, H. Reijers, A. Weijters, B. Dongen, A. Medeiros, M. Song, and H. Verbeek, "Business process mining: An industrial application," *Information Systems*, vol. 32, no. 5, pp. 713–732, 2007.

[43] M. Ehrig, A. Koschmider, and A. Oberweis, "Measuring similarity between semantic business process models," in *Conceptual Modelling 2007, Proceedings of the Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM 2007)*, J. Roddick and A. Hinze, Eds., vol. 67. Ballarat, Victoria, Australia: Australian Computer Science Communications, 2007, pp. 71–80.

[44] G. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.