Measuring Similarity between Business Process Models

Boudewijn van Dongen[1], Remco Dijkman[1], and Jan Mendling[2]

[1] Eindhoven University of Technology, The Netherlands.
{b.f.v.dongen,r.m.dijkman}@tue.nl
[2] Queensland University of Technology, Brisbane, Australia.
j.mendling@qut.edu.au

Abstract. Quality aspects become increasingly important when business process modeling is used in a large-scale enterprise setting. In order to facilitate a storage without redundancy and an efficient retrieval of relevant process models in model databases it is required to develop a theoretical understanding of how a degree of behavioral similarity can be defined. In this paper we address this challenge in a novel way. We use *causal footprints* as an abstract representation of the behavior captured by a process model, since they allow us to compare models defined in both formal modeling languages like Petri nets and informal ones like EPCs. Based on the causal footprint derived from two models we calculate their similarity based on the established vector space model from information retrieval. We validate this concept with an experiment using the SAP Reference Model and an implementation in the ProM framework.

Keywords: Business Process Modeling, Event-driven Process Chains, Similarity, Equivalence

1 Introduction

Many multi-national companies use tools such as ARIS Toolset for documenting their business processes. Due to the operational diversity of such large enterprises, there are often several thousands of processes modeled and stored in the database of the modeling tool [26]. The sheer number causes serious problems for the management and maintenance of these model: It is difficult to see the forest because there are too many trees, as a German proverb puts it. While quality aspects of process models (e.g. [15]) and process modeling languages (e.g. [10]) are quite well understood, there is a notable research gap on quality issues across models.

The similarity between business process models can be related to several of these cross-model quality issues. Consider a large organization that wants to identify redundancies in the operations of different divisions. Models are indeed helpful to discuss the overlap of two processes and the potential for integration, yet it is difficult and time-consuming to identify similarities in a process database with several thousands of models. Clearly, there is a need for automatic detection of similarities between process models to facilitate certain model management activities. There are several model management activities that would benefit from good tool support. Firstly, similar models as well as the corresponding business operations can be integrated into one process. This is interesting not only for refactoring the model database, but also to facilitate the

integration of business operations in a merger scenario. Secondly, the reference models of an ERP system vendor could be automatically compared to company processes. This way, organizations could more easily decide which packages match their current operations best. Thirdly, multi-national enterprises can identify specialized processes of some national branch which no longer comply with the procedures defined in the company-wide reference model using a similarity measurement.

In this paper, we discuss the foundations of detecting and measuring similarity between business process models. In particular, our contribution is an approach considering linguistic and behavioral aspects of process models to calculate a degree of similarity. We validate the approach using the SAP reference model. The results highlight which benefits organizations can have from tool support for similarity detection.

The remainder of the paper is organized as follows. Section 2 introduces Event-driven Process Chains (EPCs), a popular process modeling language that we use to illustrate our approach. Furthermore, we discuss one particular redundancy problem that was identified in the SAP reference model in prior research. Section 3 then presents our approach to calculate the degree of similarity between two processes based on their causal footprint. A causal footprint covers extensive behavioral information about a process without calculating its state space, but requires the identification of matching functions in the EPCs being compared. Section 4 addresses the problem of matching functions across different processes, with an emphasis on EPCs. We discuss an approach to identify matches between functions automatically. In Section 5, the presented techniques are combined, applied to a large portion of the SAP reference model, and empirically validated against human interpretations of similarity. Then, Section 6 discusses related work to our approach before Section 7 concludes the paper.

2 Background on EPCs

In this paper, we will illustrate our argument using Event-driven Process Chains (EPCs). The EPC is a popular business process modeling language that was introduced in [13]. EPCs are used by most companies that manage their process models with ARIS Toolset. This way, our results are directly applicable for these organizations.

EPCs capture the control flow of a process in terms of the temporal and logical dependencies of activities [13]. EPCs offer *function type* elements to represent these activities, *event type* elements describing pre- and post-conditions of functions, and three kinds of *connector types* including AND, OR, and XOR. Control flow arcs are used to link these elements. Connectors have either multiple incoming and one outgoing arc (join connectors) or one incoming and multiple outgoing arcs (split connectors). As a syntax rule, functions and events have to alternate on each path through the EPC, either directly or indirectly when they are linked via one or more connectors.

The informal (or intended) semantics of an EPC can be described as follows. The AND-split activates all subsequent branches in a concurrent manner. The XOR-split represents a choice between one of several alternative branches. The OR-split triggers one, two or up to all of multiple branches based on conditions. For both XOR-splits and OR-splits, the activation conditions are given in events subsequent to the connector. The AND-join waits for all incoming branches to complete, then it propagates control

to the subsequent EPC element. The XOR-join merges alternative branches. The OR-join synchronizes all active incoming branches. This feature is called non-locality since the state of all transitive predecessor nodes has to be considered. For a recent discussion of formal semantics of EPCs refer to [18].

The following definition formalizes EPC. We need this definition in the section on behavioral similarity. Furthermore, we define a notion of syntactical correctness that we check before applying our approach to the SAP reference model.

Definition 2.1. (EPC)
An $EPC = (E, F, C, l, A)$ consists of three pairwise disjoint and finite sets E, F, C, a mapping $l : C \rightarrow \{and, or, xor\}$, and a binary relation $A \subseteq (E \cup F \cup C) \times (E \cup F \cup C)$ such that

- An element of E is called *event*. $E \neq \emptyset$.
- An element of F is called *function*. $F \neq \emptyset$.
- An element of C is called *connector*.
- The mapping l specifies the type of a connector $c \in C$ as and, or, or xor.
- The relation A defines the control flow as a coherent, directed graph. An element of A is called an *arc*. An element of the union $N = E \cup F \cup C$ is called a *node*.

In order to be able to discuss the events surrounding a function, or the functions surrounding an event, notations are introduced for paths and connector chains.

Definition 2.2. (Paths and Connector Chains)
Let N be a set of $nodes$ and $A \subseteq N \times N$ a binary relation over N defining the arcs. 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 $a : a \hookrightarrow a$. If $a \neq b \in N$ and $n_2, \ldots, n_{k-1} \in C$, the path $a \overset{c}{\hookrightarrow} b$ is called *connector chain*. This includes the empty connector chain, i.e., $a \overset{c}{\hookrightarrow} b$ if $(a, b) \in A$.

In this paper, we focus on syntactically correct EPCs, i.e. EPCs with at least one initial and final events, at least one function and strict alternation of functions and events on all paths. According to this definition, both example EPCs of Figure 1 are syntactically correct. Therefore, we can apply the techniques for matching functions that are discussed later in Section 4. Out of the 604 EPCs in the SAP reference model mentioned before, 556 are syntactically correct. Please note that we demand a strict alternation of functions and events, which is not included in all EPC syntax definitions.

Figure 1 gives an example of two EPCs that captures similar processes (cf. [19]). Both are taken from the aforementioned SAP Reference Model. The EPC on the left-hand side of Figure 1 stems from the Sales and Distribution branch and its name is *Customer Inquiry*. In essence, when a customer inquires about a product (denoted by the event "Customer inquires about products"), this inquiry is processed and a quotation is created which results in the fact that a customer project is needed. As an alternative, the need for a customer project can arise based on plan data which triggers a resource related quotation. The EPC on the right-hand side of Figure 1 is taken from the Project Management branch and it is called *Customer Inquiry and Quotation Processing*. It

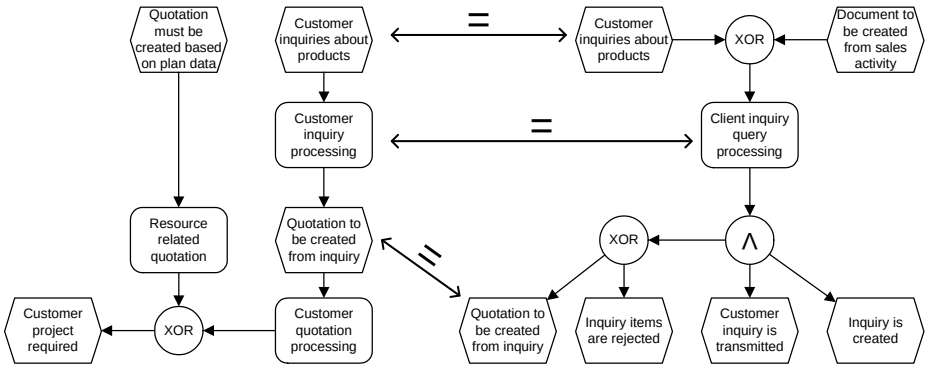

Fig. 1. *Customer Inquiry* and *Customer Inquiry and Quotation Processing* EPCs.

identifies a sales activity as alternative reason to process a customer inquiry. As a result the inquiry is created and transmitted. Furthermore, either a quotation is created or the inquiry is rejected. The processes share two equivalent events and one equivalent function as depicted in Figure 1 . Since the overlapping part of the models, i.e. the sequence "customer inquiry", "inquiry processing", and "quotation to be created", can be handled by both processes, they could easily be integrated into one model, for instance using the approach defined in [19].

In Section 3, we provide a metric for determining how similar two business processes are, given that it is known which functions (or activities in the more general sense) in one model correspond to functions in the other model. In Section 4, we show how to automatically find the relations between functions of different models.

3 Similarity of Behavior

Comparing the behavior of processes using traditional notions such as bisimulation is problematic for different reasons. Firstly, most of these notions are defined as a verification property which yield as yes or no, but no degree of similarity. Secondly, process models with concurrency suffer from a state explosion problem. For some process modeling languages a formalization of the reachability graph as a transition system is even missing. Thirdly, if there are deadlocks or dead transitions in the process model, these parts are not captured in the behavioral comparison. Motivated by these problems, we defined the concept of a causal footprint [7] which is a collection of the essential behavioral constraints imposed by a process model.[3] We will use the causal footprints of two processes as a basis to calculate their similarity. Section 3.1 describes the derivation of a causal footprint, then Section 3.2 defines the degree of similarity for causal footprints.

[3] Note that this paper adopts the concept of a causal footprint from [7] where we use it for verification purposes. In contrast to [7] we use this concept for measuring similarity.

3.1 Deriving the Causal Footprint of an EPC

Before defining a causal footprint of an EPC, we first need to introduce the notion of a case as well as the semantics of look-back and look-ahead links.

A case basically captures the behavior of one particular execution sequence of functions according to the rules of a process model. Consider N as the set of nodes of an EPC. The behavior of the process Φ_{EPC} is defined as the set $W \subseteq N^*$, where N^* is the set of all sequences that are composed of zero of more nodes from N. A $\sigma \in W$ is called a *case*, i.e. a possible execution of the EPC. To denote a function at a specific index in σ, we use $\sigma[i]$, where i is the index ranging from 1 to $|\sigma|$.

The causal footprint identifies two relationships between nodes in N that are called look-back and look-ahead links. For each *look-ahead link*, we say that the execution of the source of that link leads to the execution of at least one of the targets of that link, i.e., if $(a, B) \in L_{la}$, then any execution of a is followed by the execution of some $b \in B$. A look-ahead link is denoted as a bullet with one or more outgoing arrows. Furthermore, for each *look-back link*, the execution of the target is preceded by at least one of the sources of that link, i.e., if $(A, b) \in L_{lb}$, then any execution of b is preceded by the execution of some $a \in A$. The notation of a look-back link is a bullet with one or more incoming arrows. Note that we do not give any information about when in the future or past executions took place, but only that they are there. This way of describing a process is related to work on dominance and control dependence in program analysis (see e.g. [12]), and similar to the work presented in [8]. However, by splitting up the semantics in the two different directions (i.e. forward and backward), causal footprints are more expressive. With footprints you can for example express the fact that task `A` is always succeeded by `B`, but that `B` can also occur before `A`, which is typically hard to express in other languages.

Definition 3.1. (Causal Footprint)
We define a causal footprint $G = (N, L_{lb}, L_{la})$ as a graph where, where:

- N is a finite set of *nodes* (activities),
- $L_{lb} \subseteq (\mathcal{P}(N) \times N)$ is a set of *look-back links*[4]
- $L_{la} \subseteq (N \times \mathcal{P}(N))$ is a set of *look-ahead links*.

For relating the definition of a causal footprint to the behavior of an EPC we define a notion of consistency based on the cases implied by the EPC process model.

Definition 3.2. (Consistency of Causal Footprint with EPC)
Let N be a set of nodes and $EPC = (E, F, C, l, A)$ be an EPC with behavior W. Furthermore, let $G = (N, L_{lb}, L_{la})$ be a causal footprint. We say that $G = (N, L_{lb}, L_{la})$ is consistent with the behavior of EPC, denoted by $G \in \mathcal{F}_{EPC}$, if and only if:

1. $N = F$, i.e. the nodes of the footprint represent the functions of the EPC,
2. 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 $j : i < j \leq -1$, such that $\sigma[j] \in B$,
3. 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 $\sigma[j] \in A$,

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

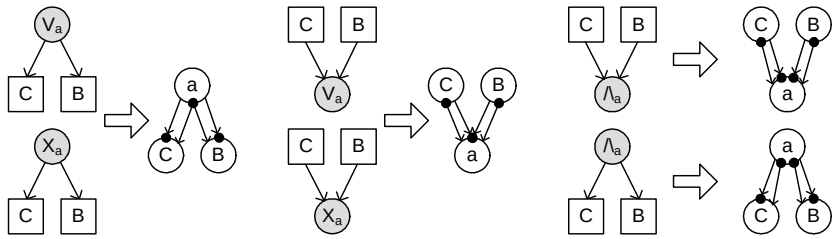

Fig. 2. Mapping of EPCs to causal footprints.

While the different cases of an EPC can explicitly be generated using the semantics formalization defined in [18], there is a more efficient way. The mapping defined in [7] and depicted in Figure 2 yields a consistent causal footprint for an EPC under the assumption that no AND-join or OR-join deadlocks. Furthermore, it is clear from Definition 3.2 that a causal footprint is not unique, i.e., different processes can have common footprints. For example, $G = (N, \emptyset, \emptyset)$ is the causal footprint of any process having activities F. Therefore, we aim at footprints that are more informative without trying to capture detailed semantics. In [7] a set of rules for calculating the transitive closure of a causal footprint are introduced such that the closure is still a causal footprint that is consistent with the EPC. In Section 5, where we present the application to the SAP reference model, we used the rules of Figure 2 in combination with the transitive closure rules of [7] to obtain a causal footprint for all EPCs.

3.2 Similarity of Causal Footprints

In information retrieval the degree of similarity between a document and a query plays a very important role for ranking the returned documents according to their relevance. For calculating similarity, we use the well-known *vector model* [2, 28] which is one of the basic techniques used for information filtering, information retrieval, and the indexing of web pages. Its classical application is to determine the similarity between a query and a document. The original vector space model proposed by Salton, Wong, and Yang in [28] attaches weights based on term frequency to the so-called "document vector". We use a more liberal interpretation, where other weights are possible. However, to explain the basic mechanism we use terms originating from the domain of information retrieval, i.e., terms like "document collection", a set of "terms", and a set of "weights" relating to the terms. Later we will provide a mapping of these terms to causal footprints.

The *document collection* contains a set of documents. Each of these documents is considered to be a list of *terms* which are basically the words of the document. The union of all terms of all documents is then used to describe each document as a vector. For one specific document an entry in the vector represents that the term associated with the vector position of this entry is included in the document. In a simple case the occurrence of a term can be indicated by a one and the non-occurrence with a zero, however there is also the option to assign *weights* to terms in order to address the fact that they differ in relevance. A common choice is to use one divided by the number of occurrences of a term throughout all documents of the document collection as a weight

which has the effect that scarcely used terms get a higher weight. A query can also be considered as a document, i.e., a list of terms.

The similarity between a query and a document is then calculated based on their vector representation as the cosine of the angle between the two vectors [2, 28]. Calculating this degree of similarity for each document provides a mechanism to rank them according to their relevance for the query.

Our proposal for determining the similarity of two business process models builds on the vector model and causal footprints. We consider causal footprints of two processes $G_1 = (N_1, L_{1,lb}, L_{1,la})$ and $G_2 = (N_2, L_{2,lb}, L_{2,la})$ as input for the calculation. In order to apply the vector model, we have to define (1) the document collection, (2) the set of terms, and (3) the set of weights.

The document collection includes two entries, namely the two causal footprints that need to be compared. We will refer to the first and the second causal footprint as $G_1 = (N_1, L_{1,lb}, L_{1,la})$ and $G_2 = (N_2, L_{2,lb}, L_{2,la})$).

The set of terms is build from the union over nodes, look back, and look ahead links of the two causal footprints. We define $\Theta = N_1 \cup L_{1,lb} \cup L_{1,la} \cup N_2 \cup L_{2,lb} \cup L_{2,la}$ as the set of terms and $\lambda : \Theta \to \{1, 2, \ldots |\Theta|\}$ as an indexing function that assigns a running number to each term, i.e., the set of all elements appearing in the two footprints are enumerated. (Note that we implicitly assume all sets of nodes and links to be disjoint in a single model.)

The relevance of each term is closely related to the number of tasks from which it is built. Consider for example two look ahead links $x_{la} = (a, \{g\}) \in L_{la}$ and $y_{la} = (a, \{b, c, d, e, f\}) \in L_{la}$. x_{la} refers to only two tasks: a and g. y_{la} refers to six tasks (a through f). It seems obvious that the look ahead links with fewer tasks are more informative and therefore more important. To address this we use weights depending on the number of tasks involved in a look-ahead/back link.

The weights are determined using the size of the relations. If $\theta \in \Theta$ is a single node (i.e. $\theta \in N_1 \cup N_2$), then we define the weight of θ as $w_\theta = 1$. Furthermore, since the number of potential look ahead and look back links depends upon the powerset of nodes, is seems natural to use exponentially decreasing weights. Therefore, for all links $\theta \in \Theta$, we define the weight of a link $w_\theta = 1/(2^{|\theta|-1})$, where $|\theta|$ denotes the number of tasks in the link.

For the two look ahead links $x_{la} = (a, \{g\})$ and $y_{la} = (a, \{b, c, d, e, f\})$, we get $w_{x_{la}} = 1/(2^{2-1}) = 0.5$ and $w_{y_{la}} = 1/(2^{6-1}) = 0.03125$ as their weights.

Using the document collection, the set of terms and the weights presented above, we define the document vectors, which we call *footprint vectors*.

Definition 3.3. (Footprint vectors)
Let $G_1 = (N_1, L_{1,lb}, L_{1,la})$ and $G_2 = (N_2, L_{2,lb}, L_{2,la})$ be two causal footprints, with Θ the set of terms and $\lambda : \Theta \to \mathbb{N}$ an indexing function. We define two footprint 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 models as follows. For each element $\theta \in \Theta$, we say that for each $i \in \{1, 2\}$ holds that

$$g_{i,\lambda(\theta)} = \begin{cases} 0 & \text{if } \theta \notin (N_i \cup L_{i,lb} \cup L_{i,la}) \\ w_\theta = \dfrac{1}{2^{|\theta|-1}} & \text{if } \theta \in (L_{i,lb} \cup L_{i,la}) \\ w_\theta = 1 & \text{if } \theta \in N_i \end{cases}$$

Using the two footprint vectors, we can define the similarity between two footprints as the cosine of the angle between these two vectors.

Definition 3.4. (Footprint similarity)
Let $G_1 = (N_1, L_{1,lb}, L_{1,la})$ and $G_2 = (N_2, L_{2,lb}, L_{2,la})$ be two causal footprints, with Θ the set of terms and $\lambda : \Theta \rightarrow \mathbb{N}$ an indexing function. Furthermore, let $\overrightarrow{g_1}$ and $\overrightarrow{g_2}$ be the corresponding footprint vectors. We say that the similarity between G_1 and G_2, denoted by $sim(G_1, G_2)$ is the cosine of the angle between those vectors, i.e.

$$sim(G_1, G_2) = \frac{\overrightarrow{g_1} \times \overrightarrow{g_2}}{|\overrightarrow{g_1}| \cdot |\overrightarrow{g_2}|} = \frac{\sum_{j=1}^{|\Theta|} g_{1,j} \cdot g_{2,j}}{\sqrt{\sum_{j=1}^{|\Theta|} g_{1,j}^2} \cdot \sqrt{\sum_{j=1}^{|\Theta|} g_{2,j}^2}}$$

The value of $sim(G_1, G_2)$ ranges from 0 (no similarity) to 1 (equivalence). In this paper, we do not elaborate on this formula. If one accepts the weights that we associate to the "terms" in a causal footprint, then the cosine of the angle between these two vectors provides a generally accepted way to quantify similarity [2, 28].

The similarity $sim(G_1, G_2)$ between footprints can be calculated for any two footprints G_1 and G_2. However, for the similarity to exceed 0, there should be at least one node $n \in N_1 \cap N_2$.

Property 3.5. (Disjoint footprints have similarity 0)
Let $G_1 = (N_1, L_{1,lb}, L_{1,la})$ and $G_2 = (N_2, L_{2,lb}, L_{2,la})$ be two causal footprints, with Θ the set of terms and $\lambda : \Theta \rightarrow \mathbb{N}$ an indexing function. Furthermore, let $\overrightarrow{g_1}$ and $\overrightarrow{g_2}$ be the corresponding footprint vectors. If $N_1 \cap N_2 = \emptyset$ then $sim(G_1, G_2) = 0$.

Proof. It is sufficient to show that $\overrightarrow{g_1} \times \overrightarrow{g_2} = 0$, i.e. that $\sum_{j=1}^{|\Theta|} g_{1,j} \cdot g_{2,j} = 0$. Assume that for some $1 \leq j \leq |\Theta|$ holds that $g_{1,j} > 0$. Then, from Definition 3.3, we know that $\lambda(\theta) = j$ with either $\theta \in N_1$, or $\theta \in (L_{1,lb} \cup L_{1,la})$. Assume $\theta \in N_1$. Then we know that $g_{2,j} = 0$, since $\theta \notin N_i$. Hence $g_{1,j} \cdot g_{2,j} = 0$. Assume $\theta \in (L_{1,lb} \cup L_{1,la})$. Since Definition 3.1 shows that $L_{1,lb} \subseteq (\mathcal{P}(N_1) \times N_1)$ and $L_{1,la} \subseteq (N_1 \times \mathcal{P}(N_1))$, we know that $\theta \notin (L_{2,lb} \cup L_{2,la})$ and hence that $g_{1,j} \cdot g_{2,j} = 0$. Therefore, $\sum_{j=1}^{|\Theta|} g_{1,j} \cdot g_{2,j} = 0$ and hence $sim(G_1, G_2) = 0$. □

Property 3.5 shows that for two footprints to be considered similar, we need to identify nodes that appear in both footprints. For this, we use the notion of an equivalence mapping defined in Section 4.

4 Matching Functions

When comparing EPCs it is not realistic to assume that equivalent functions and events have labels that are the same to the letter. Figure 1 illustrates this: the functions "Customer inquiry processing" and "Client inquiry query processing" are similar from a human perspective, but they have different labels.

To determine the match between functions from different EPCs, we:

1. determine how similar pairs of functions are on a 0 to 1 scale, based on the equivalence of words in their labels (we call this the semantic similarity score);
2. determine whether a function matches another function on a true/false scale, based on the semantic similarity score;
3. determine what the best mapping is between all functions from one EPC and all functions from another, based on the semantic similarity score; and
4. extend this technique by determining the best match by not only looking at the semantic similarity score of the functions themselves, but also at the semantic similarity scores of the events that surround these functions (we call this the contextual similarity score).

These techniques are explained successively in the following subsections.

We experimented with other techniques for determining function mappings, inspired by the work of Ehrig, Koschmider and Oberweis [9]. We also experimented with different parameters for these techniques. However, we obtained the best results for the techniques and parameters explained below. A comparison is presented in the technical report that accompanies this paper [6].

4.1 Determine the semantic similarity score between two functions

Given two functions, their semantic similarity score is the degree of similarity, based on equivalence between words in their labels. Words that are identical are given an equivalence score of 1, while words that are synonymous are given an equivalence score of 0.75, a value that was determined experimentally. We assume an exact match is preferred over a match on synonyms. Hence, the semantic similarity score is defined as follows.

Definition 4.1. (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. Let $f_1 \in F_1$ and $f_2 \in F_2$ be two functions (and assume that f_1 and f_2 are sets of words, i.e. we denote the number of words by $|f_1|$). We define the *semantic similarity* as follows:

$$\text{sem}(f_1, f_2) = \frac{1.0 \cdot |f1 \cap f2| + 0.75 \cdot \sum_{(s,l) \in f_1 \setminus f_2 \times f_2 \setminus f_1} \text{synonym}(s, l)}{\max(|f_1|, |f_2|)}$$

Where $synonym$ is a function that returns 1 if the given words are synonyms and 0 if they aren't.

For example, consider the functions "Customer inquiry processing" and "Client inquiry query processing" from figure 1, which consist of the collections of words $f1$ =["Customer","inquiry","processing"] and $f2$ =["Client", "inquiry", "query", "processing"], respectively. We only need to consider a synonym mapping between $f_1 \setminus f_2$ and $f_2 \setminus f_1$, i.e. between ["Customer"] and ["Client","query"]. Therefore, the semantic similarity between f_1 and f_2 equals
$sem(f_1, f_2) = \frac{1.0 \cdot 2 + 0.75 \cdot (1+0)}{4} \approx 0.69$.

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”. Also we *stem* words using Porter’s stemming algorithm [23]. Stemming reduces words to their stem form. For example, “stemming”, “stemmed” and “stemmer” are stemmed into “stem”.

4.2 Determine a semantic match between two functions

The semantic similarity score of two functions is a value between 0 and 1. However, when determining equivalence, we require a boolean result stating whether or not two functions are equivalent, i.e. we need cut-off values that state when the similarity score exceeds this value then the functions are equivalent. The optimal cut-off value is the cut-off value for which the syntactic similarity degree most accurately reflects the equivalence judgements of a human.

We conducted experiments to optimize these cut-off values for use in the context of the SAP Reference Models. In particular, we compared the semantic similarity scores with human judgement for 210 function pairs from the SAP Reference Model. Their similarity degrees were evenly distributed over the 0 to 1 range and they were compared against human judgement as to whether these function pairs are equivalent of not. Based on this experiment, we determined an optimal cut-off for the similarity scores to decide whether functions match or not. We expect that these cut-off values and correctness score are typical for the SAP reference model, since other data-sets yield different values [9].

Our experiments determined that for semantic similarity, a cut-off value of 0.89 while giving synonyms a similarity score higher than 0.75 is optimal. It leads to a prediction of whether functions are a match according to humans, with a 90% accuracy.

4.3 Determine a semantic mapping between all functions

So far, we only considered the similarity between two functions. However, the behavioral comparison presented in Section 3 requires a symmetric mapping between functions of two process models, i.e. we have to select pairs of functions that we consider a match, where each pair consists of a function from one model and a function from the other model.

Definition 4.2. (Equivalence mapping)
Let F_1, F_2 be two disjoint sets. Furthermore, let $s : F_1 \times F_2 \rightarrow \{0..1\}$ be a symmetric similarity function and let $c \in \{0..1\}$ be a cut-off value. A function $m : F_1 \rightarrow F_2$ is an *equivalence mapping*, if and only if:

- m is invertible ($m(f_1) = f_2$ implies that $m(f_2) = f_1$), and
- $m(f_1) = f_2$ implies that $s(f1, f2) \geq c$.

In the following section, we evaluate the degree of similarity calculation for the SAP Reference Model with different approaches to matching functions.
An *optimal equivalence mapping* $m^{opt} : F_1 \rightarrow F_2$ is an equivalence mapping, such that for all other equivalence mappings m holds that
$\sum_{(f1,f2)\in m^{opt}} s(f1, f2) \geq \sum_{(f1,f2)\in m} s(f1, f2)$.

When determining an equivalence mapping between the functions of two EPCs, each mapping satisfying Definition 4.2 is a good mapping, i.e. each element of the mapping satisfies the criterium that the similarity between the two functions exceeds the cut-off value. However, many equivalence mappings are possible. Therefore, we define the concept of an optimal equivalence mapping m^{opt}, i.e. the sum of the similarities expressed by m^{opt} is greater than the sum of the similarities of all other possible equivalence mappings[5]. An optimal equivalence mapping can be calculated in a straightforward way using integer linear programming techniques with binary variables.

4.4 Contextual Similarity

The techniques that we provided so far can be applied when comparing any two business process models. However, we are specifically considering EPCs, where each function has a preset and a postset of events. We define a second similarity metric based on this pre- and postset, which we call the contextual similarity metric. This metric produces better results than the semantic similarity metric.

Given two functions the contextual similarity technique returns the degree of similarity, based on the similarity of the events that precede and succeed them. We call these input and output events the input and output context of a function, respectively.

Definition 4.3. (Input and output context)
Let (E, F, C, l, A) be an EPC. For a function $f \in F$, we define the input context $f^{in} = \{e \in E \mid e \overset{c}{\hookrightarrow} f\}$ and the output context $f^{out} = \{e \in E \mid f \overset{c}{\hookrightarrow} e\}$

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

Definition 4.4. (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 $f_1 \in F_1$ and $f_2 \in F_2$ be two functions. Furthermore, let $m_{in}^{opt} : f_1^{in} \rightarrow f_2^{in}$ and $m_{out}^{opt} : f_1^{out} \rightarrow f_2^{out}$ be equivalence mappings between the input and output contexts of f_1 and f_2 respectively. We define the contextual similarity as follows:

$$con(f_1, f_2) = \frac{|\{m_{in}^{opt}\}|}{2 \cdot \sqrt{|f_1^{in}|} \cdot \sqrt{|f_2^{in}|}} + \frac{|\{m_{out}^{opt}\}|}{2 \cdot \sqrt{|f_1^{out}|} \cdot \sqrt{|f_2^{out}|}}$$

A full implementation of the function matching and the similarity degree calculation is available in the Process Mining framework ProM, which can freely be downloaded from www.processmining.org. In the following section we evaluate our approach using the data generated by this tool.

[5] Note that there might be more optimal equivalence mappings, however they all express a good mapping and we have no way of distinguishing between them, so any optimal equivalence mapping will suffice

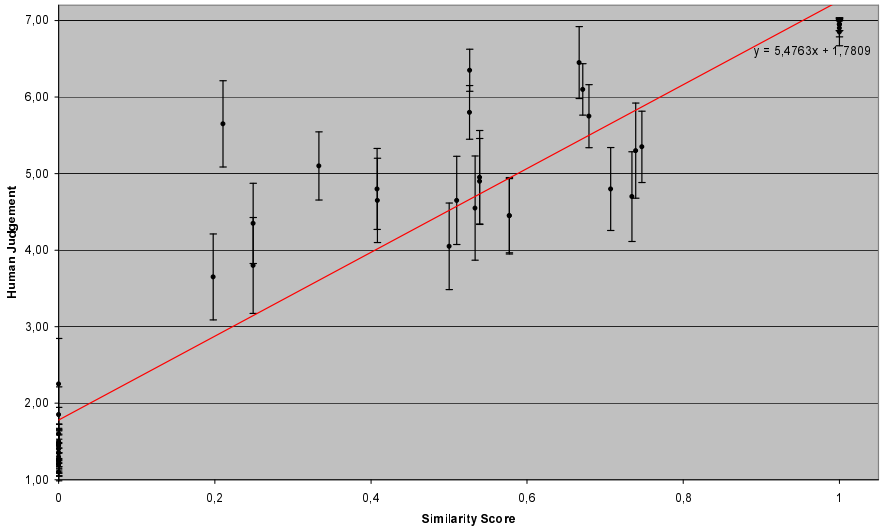

Fig. 3. Correlation between Similarity Score and Human Judgement.

5 Empirical Validation

We validated our approach to calculate the degree of similarity by computing its correlation with a similarity assessment of process modelers.

We obtained the similarity assessment using an online questionnaire that was distributed among academic process modelers. This questionnaire consisted of 48 pairs of process models from the SAP reference model database. For each pair of models, we asked the participants whether they agreed or disagreed (on a 1 to 7 Likert scale) with the proposition: 'These processes are similar.' To obtain a representative collection of model pairs, we selected the model pairs to be evenly distributed over the 0 to 1 similarity degree range. More details on how a representative collection of processes was obtained is described in the technical report that accompanies this paper [6].

We computed the correlation of the human assessment with various similarity degree metrics, which we obtained by varying cut-off values and relative importance of the syntactic, semantic and contextual similarity. We observed the best correlation for a similarity score metric that:

- does not consider syntactic similarity,
- uses a cut-off value of 0.89 for semantic similarity of events,
- uses a relative importance of semantic:contextual similarity of 1:2 and a cut-off value of 0.90 for similarity of functions.

Figure 3 shows the correlation between the similarity degree (computed using the settings described above) and the similarity assessment as obtained from the questionnaire. Each point in the graph represent a pair of processes, with a similarity degree as indicated by its x-value and a human similarity assessment as indicated by its y-value. The confidence intervals are also plotted (with a 90% confidence). For this metric we

got a high (Pearson) correlation coefficient of 0.84 with the human judgement. The correlation is represented as a straight line in the graph. The correlation for two other metrics that we investigated was lower, i.e. the metric presented here was the best one. Details on all similarity degree metrics are given in the technical report that accompanies this paper [6].

An important observation is that, within the 'sales and distribution' branch of the SAP reference model (which contains 74 models), there are 124 process pairs with a similarity score of 1 (this is 50 more than the expected 74 pairs that represent comparison of a process with itself). In addition to that there are 52 process pairs with a similarity score s, such that $0.5 \leq s < 1.0$. These figures show the overlap between processes in 'sales and distribution' branch. This information can be used by people that are searching the SAP reference model for a suitable process; they can find overlapping processes based on this information. It can also be used to maintain consistency when updating a process for which there exists an overlapping process.

6 Related Work

This paper mainly relates to two streams of research, namely (1) similarity of business process models and (2) quality of business process models.

Existing work in the context of determining *similarity* between process models can be assigned to three categories: verification, behavioral similarity, and textual similarity. There are different notions of equivalence of process models that are subject to *verification* such as trace equivalence and bisimulation. While trace equivalence is based on a comparison of the sets of completed execution traces, bisimulation also considers at which point of time which decisions are taken, i.e., bisimulation is a stricter notion of equivalence. Details on different equivalence notions are given e.g. in [1]. A general problem of such verification approaches is that it provides a true-false answer to the question whether two models are similar. While some work has been done on determining a degree of behavioral similarity that measures the fitness of a set of event logs relative to a process model [1], we compare causal footprints [7] of two process models. Since causal footprints capture constraints instead of the state space, this approach relates to declarative approaches to process modeling and verification [8,17,22]. Beyond that, there are some works on textual or metadata similarity of process models (e.g. [9,14,20]). In this paper we adapt some concepts from this area for matching function labels, and we combine this approach with the calculation of behavioral similarity.

While there has been intensive research into quality aspects of process models and process modeling languages [3, 10, 15], there is little work on quality issues across models. The guidelines of modeling [3] touch this area by stressing the importance of a systematic design. The novelty of our approach is that systematic design in terms of non-overlapping models can now be checked automatically. This might prove valuable for providing tool support for process model normalization as defined in [21]. Beyond that, the quantification of a degree of behavioral similarity between process models could be a useful contribution for the area of process model *integration*. While there are several approaches reported on integration issues [5] and regarding *how* two models are integrated (e.g. [11,19,24]) the similarity degree gives an answer to the question *which*

two process models might be good candidates for integration, e.g. in a merger situation. The redundancies that we identified in the SAP reference model underline the need for techniques and tools to manage process model variants such as defined in [25, 27]. Furthermore, there is clearly a need for a view concept on business process models in order to avoid anomalies [4] as they were identified in database research before.

7 Conclusion

In this paper, we presented a novel approach for measuring the degree of similarity of business process models. This approach builds on the vector model from information retrieval, an abstract representation of process behavior as causal footprints, and an automatic matching of functions across process models. While quality aspects of single process models and process modeling languages are well understood, this work contributes to a better foundation of those quality aspects across models that relate to similarity. Our approach has been validated using the SAP Reference Model, and a respective implementation is available as part of the ProM framework.

The results that we obtained for the SAP Reference Model clearly highlight the need for an automatic detection of similarity for supporting refactoring activities of a process model database. In future research we will investigate the benefits of our approach in various case studies. In particular, we aim to use the degree of similarity to detect operational overlap between companies that engage in a merger. While the application for the SAP Reference Model could build on a presumably homogeneous vocabulary of function labels, we assume that synonyms in function labels might play a more important role in a merger. Furthermore, there are some practical issues with reading the similarity matrix for a large set of models that need to be addressed. Once there is commercial tool support available, companies will find it easier to maintain large databases of process models.

References

1. W.M.P. van der Aalst, Ana Karla Alves de Medeiros, and A.J.M.M. Weijters. Process Equivalence: Comparing two process models based on observed behavior. In S. Dustdar, J.L. Fiadeiro, and A. Sheth, editors, *Proc. of BPM 2006*, LNCS 4102, pages 129–144, 2006.
2. R.A. Baeza-Yates and B.A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.
3. J. Becker, M. Rosemann, and C. von Uthmann. Guidelines of Business Process Modeling. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management. Models, Techniques, and Empirical Studies*, pages 30–49. Springer, Berlin et al., 2000.
4. J. Biskup. Achievements of relational database schema design theory revisited. In L. Libkin, B. Thalheim, editor, *Semantics in Databases*, LNCS 1358, pages 29–54, 1998.
5. R. Dijkman. A Classification of Differences between Similar Business Processes. In *Proceedings of the 11th IEEE EDOC Conference (EDOC'07)*, pages 37–50, 2007.
6. B.F. van Dongen, R.M. Dijkman, J. Mendling. Detection of similarity between business process models. BETA Working Paper 233, Eindhoven University of Technology, 2007.
7. B.F. van Dongen, J. Mendling, and W.M.P. van der Aalst. Structural Patterns for Soundness of Business Process Models. In *Proceedings of the 10th IEEE International EDOC Conference (EDOC'06)*, pages 116–128, 2006. IEEE.

8. H. Eertink, W. Janssen, P. Oude Luttighuis, W.B. Teeuw, and C.A. Vissers. A business process design language. In J.M. Wing, J. Woodcock, and J. Davies, editors, *World Congress on Formal Methods*, LNCS 1708, pages 76–95, 1999.
9. M. Ehrig, A. Koschmider, and A. Oberweis. Measuring similarity between semantic business process models. In J.F. Roddick and A. Hinze, editors, *Proceedings of the Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM 2007)*, pages 71–80, 2007.
10. P. Green and M. Rosemann. Integrated Process Modeling. An Ontological Evaluation. *Information Systems*, 25(2):73–87, 2000.
11. G. Grossmann, Y. Ren, M. Schrefl, and M. Stumptner. Behavior based integration of composite business processes. In *Proceedings of the 3rd International Business Process Management Conference (BPM 2005)*, LNCS 3649, pages 186–204, 2005.
12. R. Johnson, D. Pearson, and K. Pingali. The program structure tree: Computing control regions in linear time. In *Proceedings of the ACM SIGPLAN'94 Conference on Programming Language Design and Implementation. SIGPLAN Notices 29(6)*, pages 171–185, 1994.
13. G. Keller, M. Nüttgens, and A.-W. Scheer. Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)". Heft 89, Institut für Wirtschaftsinformatik, Saarbrücken, Germany, 1992.
14. M. Klein and A. Bernstein. Toward high-precision service retrieval. *IEEE Internet Computing*, 8(1):30–36, 2004.
15. J. Krogstie, G. Sindre, and H.D. Jørgensen. Process models representing knowledge for action: a revised quality framework. *Europ. J. of Information Systems*, 15(1):91–102, 2006.
16. I. Levenshtein. Binary code capable of correcting deletions, insertions and reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1966.
17. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, New York, 1991.
18. J. Mendling and W.M.P. van der Aalst. Formalization and Verification of EPCs with OR-Joins Based on State and Context. In J. Krogstie, A.L. Opdahl, and G. Sindre, editors, *Proceedings of the 19th Conference on Advanced Information Systems Engineering (CAiSE 2007)*, LNCS 4495, pages 439–453, 2007.
19. J. Mendling and C. Simon. Business Process Design by View Integration. In Johann Eder and Schahram Dustdar, editors, *Proceedings of BPM Workshops 2006*, volume 4103 of *Lecture Notes in Computer Science*, pages 55–64, Vienna, Austria, 2006. Springer-Verlag.
20. M. Momotko and K. Subieta. Process query language: A way to make workflow processes more flexible. In G. Gottlob, A.A. Benczúr, and J. Demetrovics, editors, *ADBIS 2004, Proceesing*, LNCS 3255, pages 306–321, 2004.
21. V. Pankratius and W. Stucky. A formal foundation for workflow composition, workflow view definition, and workflow normalization based on petri nets. 2005.
22. M. Pesic, M.H. Schonenberg, N. Sidorova, and W.M.P. van der Aalst. Constraint-based workflow models: Change made easy. pages 77–94, 2007.
23. M F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
24. G. Preuner, S. Conrad, and M. Schrefl. View integration of behavior in object-oriented databases. *Data & Knowledge Engineering*, 36(2):153–183, 2001.
25. J. Recker, J. Mendling, M. Rosemann, and W.M.P. van der Aalst. Model-driven Enterprise Systems Configuration. In *Proceedings of the 18th Conference on Advanced Information Systems Engineering (CAiSE 2006)*, LNCS 4001, pages 369–383, 2006.
26. M. Rosemann. Potential pitfalls of process modeling: part b. *Business Process Management Journal*, 12(3):377–384, 2006.
27. M. Rosemann and Wil van der Aalst. A Configurable Reference Modelling Language. *Information Systems*, 32:1–23, 2007.
28. G. Salton, A. Wong, and C.S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620, 1975.