On the Degree of Behavioral Similarity between Business Process Models

Jan Mendling[†], Boudewijn van Dongen[‡], Wil van der Aalst[‡]
[†]Queensland University of Technology
126 Margaret Street, QLD 4000 Brisbane, Australia
j.mendling@qut.edu.au
[‡]Eindhoven University of Technology
PO Box 513, NL-5600 MB Eindhoven, The Netherlands
b.f.v.dongen@tue.nl, w.m.p.v.d.aalst@tue.nl

Abstract: Quality aspects become increasingly important while 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 illustrate this concept with an example from the SAP Reference Model and present a prototypical implementation as a plug-in to the ProM framework.

1 Introduction

Business process modeling is gaining increasing attention as a basis for the development of large-scale enterprise information systems. In this context, business process models can either be used as a formalization of requirements that guide the implementation, as input for code generation in a model-driven architecture, or as executable templates on a dedicated process engine defined e.g. with BPEL. All these three scenarios have in common that various quality issues have to be considered in order to facilitate *storage without redundancy* and an *efficient retrieval of models.* These two goals have in common that there require a theoretical understanding of how the degree of behavioral similarity can be formalized. Based on such a concept it would be easy to identify those models that should be integrated with others for maintainability reasons and to rank models that belong to the answer set of a query. Such a functionality would be in particular helpful to analysts who have to integrate the operations of two enterprises that engage in a merger. We will see in the related work section that the current state of the art does not provide a general solution to this problem.

The analysis of behavioral similarity of business process models is complicated by two

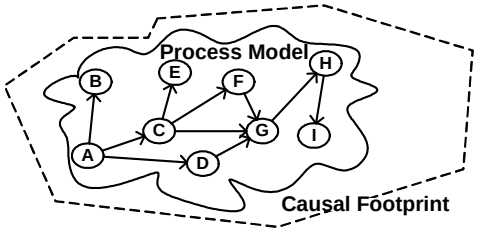

Figure 1: A causal footprint provides a characterization of the process model such that the behavior of the process is enclosed [DMA06].

problems. First, there is a plethora of languages for business process modeling. Even though there is some overlap between the different languages as shown by pattern-based evaluations [AHKB03], there are striking differences and many subtle semantical issues that complicate matters. For example, a concept such as the "synchronizing merge" [AHKB03], also know as the OR-join in languages such as BPMN and EPCs, can be interpreted in different ways and often leads to semantical paradoxes as shown in [ADK02, Kin06]. Second, and even worse, most business process modeling languages have no formal semantics or their semantics is only well-defined for a restricted subset. Still, redundancy and retrieval issues matter also for those business process models. In this paper, we address these problems in a novel way.

The classical approach to comparing process models is to construct a state space or enumerate all possible traces and then compare the models based on this. Trace equivalence and bisimulation are typical notions used to compare formal models on such basis. Unfortunately, these equivalence notions typically only work for models that have formal semantics and have finite behavior (e.g., the number of traces or states needs to be bounded). Moreover, such notions provide a "yes/no" answer rather than the degree of similarity. Therefore, we abstract from the precise behavior using a concept that is called *causal footprint* [DMA06]. The idea of a causal footprint is that it describes a set of conditions on the order of activities that hold for the process model. These conditions can be used to reason about *similarity* of footprints. Figure 1 illustrates that the causal footprint gives an approximation of the process behavior in terms of conditions that every process instance has to obey to. In [DMA06] it has been shown that causal footprints can be used to reason about the soundness of process models. Although causal footprints abstract from the detailed behavior, it is still possible to find certain errors (e.g., particular types of deadlocks).

In this paper, we use *causal footprints for measuring similarity*. Unlike our earlier work [DMA06] we do not look at verification but at the degree of similarity between two models. When comparing two models we compare the corresponding causal footprints using the so-called vector model used in information retrieval [SWY75, BYRN99].

Throughout this paper, we use Event-driven Process Chains (EPCs) and Petri nets to show the applicability of causal footprints for both formal and conceptual business process modeling languages. This choice is motivated by the fact that EPCs are widely used for the documentation of business processes, e.g. in the SAP reference model. Furthermore, Petri nets are a well-understood formalism for the modeling of business processes and

have been used for the formalization of a variety of languages and standards (e.g. BPMN, BPEL, XPDL, UML activity diagrams). Against this background the paper is structured as follows. First, Section 2 gives an introduction to EPCs and Petri nets and shows how both can be mapped to Causal Footprints. Then, Section 3 presents a novel concept for calculating the degree of behavioral similarity between two process models based on their causal footprints and inspired by concepts from information retrieval. Section 6 discusses our approach in the light of related work before Section 7 concludes the paper with an outlook on future research.

2 Preliminaries

In the introduction, we mentioned two process modeling languages, namely Petri nets and EPCs. In this section, we introduce these modeling languages informally and provide formal definitions for both languages (only syntax). Furthermore, we introduce the concept of a causal footprint and show how to derive causal footprint from EPCs and Petri nets by just considering their structure. Most modeling languages are graph based. Therefore, we start by introducing some notation specifically for directed graphs.

Definition 2.1. (Pre-set and Post-set)

Let $G = (N, E)$ be a directed graph consisting of a set of nodes N and a set of edges $E \subseteq N \times N$. Moreover, let $n \in N$ be a particular node. We define ${}^{G}\!\bullet n = \{m \in N \mid (m, n) \in E\}$ as the pre-set and $n {\bullet\!}^{G} = \{m \in N \mid (n, m) \in E\}$ as the post-set of n with respect to the graph G. If the context is clear, the superscript G may be omitted, resulting in $\bullet n$ and $n\bullet$.

2.1 Petri nets and EPCs

Petri nets can be used to specify (possibly concurrent) processes in an unambiguous manner. Since the language has formal and executable semantics, processes modelled in terms of a Petri nets can be executed by an information system. In fact most workflow engines have adopted the basic constructs present in the Petri net formalism. For an elaborate introduction to Petri nets, the reader is referred to [DE95,Mur89,RR98]. A Petri net consists of two types of node elements (cf. Figure 2): *Transitions* typically correspond to either an activity which needs to be executed, or to a "silent" step that takes care of routing. A transition is drawn as a rectangle. *Places* are used to define the preconditions and postconditions of transitions. A place is drawn as a circle. Transitions and places are connected through directed arcs in such a way that (i) places and transitions have at least one incident edge and (ii) in every path, transitions and places alternate (no place is connected to a place and no transition is connected to a transition). For completeness sake, we mention that the Petri nets we use in this paper correspond to a classic subclass of Petri nets, namely workflow nets [Aal98], which are tailored towards workflow modeling and analysis.

Definition 2.2. (Workflow net)

$\omega = (P, T, F)$ is a workflow net (or WF-net [Aal98]) if:

- P is a finite set of places,
- T is a finite, non empty set of transitions, such that $P \cap T = \emptyset$ and $T \neq \emptyset$,
- $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation of the net,
- there exists exactly one $p_i \in P$, such that $| \bullet p_i| = 0$,
- there exists exactly one $p_f \in P$, such that $|p_f \bullet | = 0$,
- all places and transitions are covered by the paths from p_i to p_f.

In addition to Petri nets, we also use EPCs to illustrate our ideas. EPCs provide an intuitive modeling language to model business processes. EPCs were introduced by Keller, Nüttgens, and Scheer in 1992 [KNS92]. It is important to realize that the language is not intended to be a *formal* specification of a business process. Instead, it serves mainly as a means of communication. EPCs are extensively used in large-scale enterprise modeling projects. One prominent example of a publicly available model is the SAP reference model [CKL97, KT98]. An EPC consists of three types of node elements (cf. Figure 2): *Functions* correspond to an activity (task, process step) which needs to be executed. A function is drawn as a box with rounded corners. *Events* describe the situation before and/or after a function is executed. An event typically corresponds to the pre- or post-condition of a function. Events are drawn as hexagons. *Connectors* can be used to connect functions and events to specify the flow of control. There are three types of connectors: $\wedge$ (AND), $\times$ (XOR) and $\vee$ (OR). Connectors are drawn as circles, showing the type in the center. Functions, events, and connectors can be connected with edges in such a way that (i) events have at most one incoming edge and at most one outgoing edge, but at least one incident edge (i.e. an incoming or an outgoing edge), (ii) functions have precisely one incoming edge and precisely one outgoing edge, (iii) connectors have either one incoming edge and multiple outgoing edges, or multiple incoming edges and one outgoing edge, and (iv) in every path, functions and events alternate (ignoring intermediate connectors).

Definition 2.3. (Event-driven Process Chain)

$\varepsilon = (F, E, C_{and}, C_{xor}, C_{or}, A)$ is an EPC if:

- F is a finite set of functions,
- E is a finite set of events,
- $C = C_{and} \cup C_{xor} \cup C_{or}$ is a finite set of connectors, such that $|C| = |C_{and}| + |C_{xor}| + |C_{or}|$,
- $A \subseteq ((F \cup E \cup C) \times (F \cup E \cup C))$ is the flow relation of the net, such that:
  - for all $f \in F$ there is one $(f, x) \in A$ and one $(x, f) \in A$,
  - for all $e \in E$ there is at most one $(e, x) \in A$ and at most one $(x, e) \in A$,
  - for all $c \in C$ there is either one $(c, x) \in A$ and more than one $(x, c) \in A$, or one $(x, c) \in A$ and more than one $(c, x) \in A$,
  - on all paths, functions and events alternate.

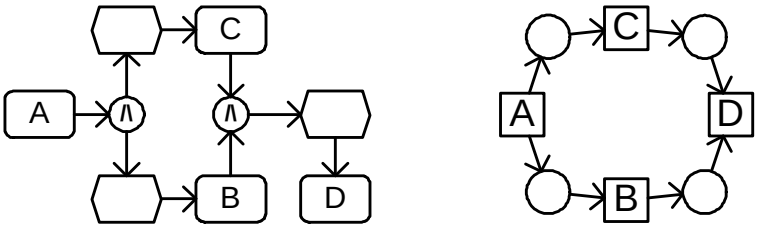

Figure 2: An EPC (left) and a Petri net (right) describing a process thats starts with A, followed by the parallel execution of B and C, and ends with D.

Note that in this section, we only provide an abstract syntax of Petri nets and EPCs and do not give any semantics. The reason is that we are not interested in the precise semantics. As illustrated by Figure 1 we aim for a causal footprint which is independent of specific and/or refined semantical interpretations (e.g., the OR-join in EPCs).

2.2 Causality graphs

As stated in the introduction, we look at a similarity measure for business processes, based on the high-level concept of causal footprints. At the basis of causal footprints lie *causality graphs*.[1] A causality graph captures the intented behavior that is captured in the process model. It is important to realize that these causality graphs do not capture the entire process model as such, in fact they are merely a footprint of the control flow in the given process model, i.e. they describe the approximate behavior of a system at a very high level.

Definition 2.4. (Process behavior/case)

Let T be a set of activities, and let Φ_T be a process containing these activities. The behavior of the process Φ_T is defined as the set $W \subseteq T^*$, where T^* is the set of all sequences that are composed of zero of more tasks from T. A $\sigma \in W$ is called a *case*, i.e. a possible execution of the process. To denote an activity at a specific index in σ, we use $\sigma[i]$, where i is the index ranging from 1 to $|\sigma|$.

We have now formalized the behavior of a process and each of the modeling languages of Section 2 is intended to capture this behavior in a structured way. As we stated before, we will look at processes at a high level. Therefore, we introduce the *causality graph*, representing a footprint of the process.

Definition 2.5. (Causality Graph)

Let N be a set of activities. We define a causality graph $G = (N, F_{lb}, F_{la})$, where:

- N is a finite set of *nodes* (activities),

[1] Note that this paper adopts the concept of a causality graph from [DMA06]. However, unlike in [DMA06] this concept is used for measuring similarity rather than verification.

- $F_{lb} \subseteq (\mathcal{P}(N) \times N)$ is a set of *look-back links*[2],
- $F_{la} \subseteq (N \times \mathcal{P}(N))$ is a set of *look-ahead links*.

When a causality graph is used to describe a process, it should be interpreted in the following way. 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 $(a, B) \in F_{la}$, 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 $(A, b) \in F_{lb}$, 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 similar to the work presented in [EJL$^+$99]. 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.

If a causality graph indeed describes a process like this, we call it a *causal footprint*. We formalize this concept using the notion of cases.

Definition 2.6. (Causal Footprint)

Let T be a set of activities, Φ_T be a process with behaviour W. Furthermore, let t_i and t_f be such that $t_i, t_f \notin T$ and $t_i \neq t_f$. Furthermore, let $G = (T \cup \{t_i, t_f\}, F_{lb}, F_{la})$ be a causality graph. (For notational purposes, we say that for all $\sigma \in W$ with $n = |\sigma|$, it holds that $\sigma[0] = t_i$ and $\sigma[n+1] = t_f$, i.e. we add artificial starts and ends to each trace). We say that G is a *causal footprint* graph of Φ_T, denoted by $G \in \mathcal{F}_{\Phi_T}$, if and only if:

1. For all $(a, B) \in F_{la}$ 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 $(A, b) \in F_{lb}$ 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 $\sigma[j] \in A$,

It is clear from Definition 2.6 that a causal footprint is not unique, i.e., different processes can have common footprints. For example, $G = (T \cup \{t_i, t_f\}, \emptyset, \emptyset)$ is the causal footprint of any process having activities T. Therefore, we aim at footprints that are more informative without trying to capture detailed semantics. By deriving a transitive closure, a causal footprint can be extended to be a more informative causal footprint.

In [DMA06], the transitive closure of a causal footprint was defined for the soundness analysis of these footprints. In this paper, we use the same *causal closure* to describe similarities between process models.

Definition 2.7. (Causal Closure)

Let $G = (N, F_{lb}, F_{la})$ be a causality graph. We define $G^* = (N, F^*_{lb}, F^*_{la})$ to be the causal closure of G, where F^*_{lb} and F^*_{la} are the smallest possible sets, such that:

[2]With $\mathcal{P}(N)$, we denote the powerset of N, i.e. $N' \in \mathcal{P}(N)$ if and only if $N' \subseteq N$ and $N' \neq \emptyset$.

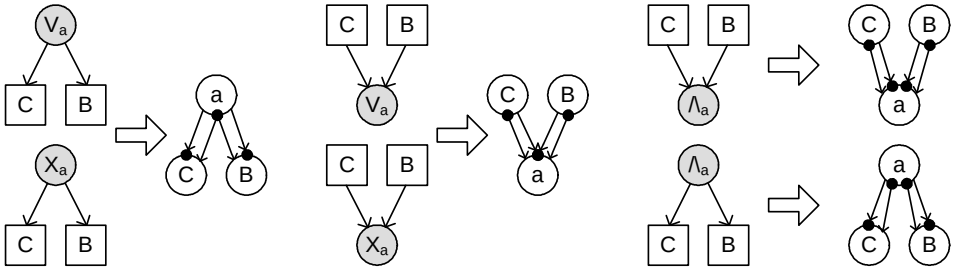

Figure 3: Mapping of EPCs to causal footprints.

1. $(a, B) \in F_{la}$ implies that $(a, B) \in F^*_{la}$,
2. $(A, b) \in F_{lb}$ implies that $(A, b) \in F^*_{lb}$,
3. $(a, B) \in F^*_{la}$ implies that for all $N' \subseteq N$ holds that $(a, B \cup N') \in F^*_{la}$,
4. $(A, b) \in F^*_{lb}$ implies that for all $N' \subseteq N$ holds that $(A \cup N', b) \in F^*_{lb}$,
5. $(a, B) \in F^*_{la}$, $b \in B$ and $(b, C) \in F^*_{la}$, implies that $(a, (B \setminus \{b\}) \cup C) \in F^*_{la}$,
6. $(B, c) \in F^*_{lb}$, $b \in B$ and $(A, b) \in F^*_{lb}$, implies that $((B \setminus \{b\}) \cup A, c) \in F^*_{lb}$,

The rules for causally closing a causal graph obviously apply to a causal footprint as well. More importantly, the causal closure of a causal footprint is a causal footprint again [DMA06] and we refer to it as a *footprint closure*. The fact that a causal footprint of a process is not unique is irrelevant for the work presented in this paper. Any property that can be derived to hold on a causal footprint of a process, holds on the process itself. The better the causal footprint of a process is, i.e. the more information it contains, the more properties we are able to deduce. As an example, again consider the graph that only has the activities of a process as nodes and no edges. This graph is a causal footprint of any process, but it does not contain any information about the process.

2.3 Deriving Causal Footprints

In this section, we present algorithms to derive causal footprints of EPCs and Petri nets. This is done in two stages. First, the models are translated to causality graphs in such a way that these graphs contain all elements of the modeling languages and only causalities that can be derived locally (i.e., immediate predecessor and successor relationships). These graphs are then transitively closed and projected onto the subset of interesting elements, thus leading to causal footprints.

Definition 2.8. (EPC to causal graph)

Let $\varepsilon = (F, E, C_{and}, C_{xor}, C_{or}, A)$ be an EPC, with the set of modeling elements $N' = F \cup E \cup C_{and} \cup C_{xor} \cup C_{or}$, the set of initial elements $N_i = \{n \in N' \mid \overset{\varepsilon}{\bullet} n = \emptyset\}$ and the set of final elements $N_f = \{n \in N' \mid n \overset{\varepsilon}{\bullet} = \emptyset\}$. We define a causality graph $G_\varepsilon = (N, F_{lb}, F_{la})$ as follows:

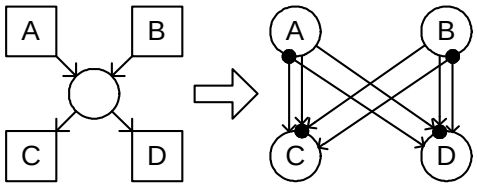

Figure 4: Mapping for Petri net.

- $N = N' \cup \{n_i, n_f\}$, where $n_i, n_f \notin N'$ and $n_i \neq n_f$,
- $F_{la} =$
  $\{(a, \{b\}) \in N \times \mathcal{P}(N) \mid a \in F \cup E \cup C_{and} \wedge b \in a\overset{\varepsilon}{\bullet}\} \cup$
  $\{(a, B) \in N \times \mathcal{P}(N) \mid a \in C_{or} \cup C_{xor} \wedge B = a\overset{\varepsilon}{\bullet}\} \cup$
  $\{(n_i, N_i)\} \cup \{(a, \{n_f\}) \mid a \in N_f\}$,
- $F_{lb} =$
  $\{(\{a\}, b) \in \mathcal{P}(N) \times N \mid b \in F \cup E \cup C_{and} \wedge a \in \overset{\varepsilon}{\bullet} b\} \cup$
  $\{(A, b) \in \mathcal{P}(N) \times N \mid b \in C_{or} \cup C_{xor} \wedge A = \overset{\varepsilon}{\bullet} b\} \cup$
  $\{(N_f, n_f)\} \cup \{(\{n_i\}, a) \mid a \in N_i\}$.

Definition 2.8 gives an algorithm to derive a causality graph from an EPC. This is illustrated by Figure 3. It is important to note that the causality graph is derived based on the structure of the model, i.e., there is no need to first construct the set of cases. In some cases it is useful to restrict the causality graph to a subset of nodes. For EPCs for example, we might only be interested in its functions and connectors. By taking the causal closure of the causality graph and projecting it onto the functions and connectors (i.e. removing all events and all edges related to these events), we obtain a causal footprint of the process modelled by the EPC. For this, we first define a projection of a causal graph.

Definition 2.9. (Causal Graph Projection)

Let $G = (N, F_{lb}, F_{la})$ be a causal graph and let N' be a set of nodes, such that $N' \subseteq N$. We define the projection $G' = (N', F'_{lb}, F'_{la})$ of G onto N', such that ${F_{lb}}' = F_{lb} \cap (\mathcal{P}(N') \times N')$ and ${F_{la}}' = F_{la} \cap (N' \times \mathcal{P}(N'))$.

As we stated before, the translation rules from an EPC to a causal graph are given under the assumption that the EPC is sound. However, soundness would imply that there are clear executable semantics, which is not always the case. Therefore, our translation rules do not require explicit semantics. The only requirement for our rules to hold is that soundness should be defined in such a way that a synchronizing connector is considered to fire eventually.

Similar to EPCs, we provide a translation for WF-nets to causality graphs and show that the result is a causal footprint for the modelled process (cf. Figure 4).

Definition 2.10. (Workflow net to causal footprint)

Let $\omega = (P, T, F)$ be a WF-net, with the set of modeling elements $N' = P \cup T$, the initial place $p_i \in P$ and final place $p_f \in P$. We derive a causality graph $G_\omega = (N, F_{lb}, F_{la})$ as

follows:

- $N = T \cup \{n_i, n_f\}$, where $n_i, n_f \notin N'$and $n_i \neq n_f$,
- $F_{la} =$
  $\{(n, N') \in N \times \mathcal{P}(N) \mid \exists_{p \in P \setminus \{p_f\}}\ n \in \overset{\omega}{\bullet} p \wedge N' = p \overset{\omega}{\bullet}\} \cup$
  $\{(n_i, p_i \overset{\omega}{\bullet})\} \cup$
  $\{(t, \{n_f\}) \mid t \in \overset{\omega}{\bullet} p_f\}$,
- $F_{lb} =$
  $\{(N', n) \in \mathcal{P}(N) \times N \mid \exists_{p \in P \setminus \{p_i\}}\ n \in p \overset{\omega}{\bullet} \wedge N' = \overset{\omega}{\bullet} p\} \cup$
  $\{(\overset{\omega}{\bullet} p_f, n_f)\} \cup$
  $\{(\{n_i\}, t) \mid t \in p_i \overset{\omega}{\bullet}\}$,

As shown in [DMA06] causal footprints can be used to detect problems such as deadlocks. However, the causal footprint also provides a compact but useful characterization of the corresponding process. Therefore, the concept can be used to compare processes which is useful when searching for similar models or for quantifying the gap between some desired model and a given model.

3 Similarity of Business Process Models

In many scenarios it is important to compare business process models with respect to the behavior they specify, e.g. in case of a merger of two companies that have models for similar business processes, or when looking for a web services closest to the own internal processes. In this section, we consider the causal footprints of two models as an input to calculate similarity. This has several advantages compared to related proposals. First, causal footprints include information about the order of activities beyond direct succession. The full closure also looks at indirect dependencies and, therefore, it is more powerful than the approach in [AAW06], since there only direct connections are used for structural similarity. Furthermore, causal footprints are robust with respect to problems such as termination or finiteness of state space which affect true/false behavioral similarity measures, such as trace-equivalence and (branching) bisimilarity. We calculate similarity directly based on the causal footprint of two models instead of considering their explicit behavior. In order to do so we use techniques originating from information retrieval [SWY75, BYRN99].

3.1 Similarity based on the Vector Model

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* [SWY75, BYRN99] 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 [SWY75] 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 [SWY75, BYRN99]. 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 $G_1 = (N_1, F_{1,lb}, F_{1,la})$ and $G_2 = (N_2, F_{2,lb}, F_{2,la})$ 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 corresponding to the two causal footprints that need to be compared. We will refer to these as the first and the second causal footprint (i.e., G_1 and G_2).

The set of terms is build from the union over nodes, look back, and look ahead links of the two causality closures. We define $\Theta = N_1 \cup F_{1,lb} \cup F_{1,la} \cup N_2 \cup F_{2,lb} \cup F_{2,la}$ 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 $x_{la} = (a, \{g\}) \in F_{la}$ and $y_{la} = (a, \{b, c, d, e, f\}) \in F_{la}$. 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, i.e. 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 $w_{y_{la}} = 1/(2^{6-1}) = 0.03125$.

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

Definition 3.1. (Footprint vectors)

Let $G_1 = (N_1, F_{1,lb}, F_{1,la})$ and $G_2 = (N_2, F_{2,lb}, F_{2,la})$ be two causal footprints, with Θ the set of terms and $\lambda : \Theta \rightarrow \mathbb{N}$ 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

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

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

Definition 3.2. (Footprint similarity)

Let $G_1 = (N_1, F_{1,lb}, F_{1,la})$ and $G_2 = (N_2, F_{2,lb}, F_{2,la})$ 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 [SWY75, BYRN99].

The similarity $sim(G_1, G_2)$ between footprints can be calculated for any two footprints G_1 and G_2. However, the best results will be obtained if the causal footprint is a footprint closure, since then the most information is obtained. To illustrate this, we again take two examples from the SAP reference model.

4 Application to the SAP reference model

To show the practical applicability in real life, we applied our approach in the context of the SAP reference model. In this section, we illustrate the calculation of business process model similarity based on the vector model and causal footprints using this reference model.

Since reference models should be comprehensive sets of models serving as a guide when modelling large information systems, we feel that they should satisfy two properties:

- The reference model should contain models that are correct, i.e. there should not be any errors in these models.
- The reference model should be searchable, i.e. one should not only be able to search for models of a specific process (such as "purchasing" or "invoice management"), but one should also be able to search through the database to find a model that matches a given model as close as possible. This is in particular needed in merger situations where similar operations of two enterprise have to be integrated.

The first requirement has been extensively discussed in [DJVVA, ADMV06, MADV06b, MADV06a, MVD+07] and it was mentioned in [DMA06], where we showed how causal footprints can be used to find errors in EPCs.

A first step towards satisfying the latter of the two requirements is provided in this paper, i.e. we present a way to compare the behaviour of models based on their structure. In the work presented here, it is assumed that it is easy to map the activities of one model to the activities of another model (e.g. by assuming that they are the same objects, or have the same label). In our application, this mapping was done manually since the labels of functions not always coincided completely (e.g. "Order creation" vs. "refurbishment order creation", etc.).

In [DJVVA], we presented a guided model selection to find errors in the SAP reference model. In that paper, we were guided by one specific function called "Order Execution" that appeared in several EPCs (seven in total). In this paper, we take the same guide, i.e. we look at the seven EPCs containing the function "Order Execution" and we pairwise compare their similarity. The seven EPCs that are investigated are presented in Table 1.

For each of the seven models, we calculated the causal footprint and we compared them using our similarity metric of Definition 3.2. It is important to note that the seven models do not all meet the first requirement stated above, i.e. most of them even show syntactical problems by having events linked to events. However, since events are not reflected in a causal footprint, this does not influence our results.

Table 2 shows the result of our comparison. Again note that at some points, we manually had to make mappings from the functions in one model to functions in the other model. Figure 5 shows how this mapping was made in the implementation in ProM (cf. Section 5). It shows that a mapping was made from "order creation" to "maintenance order creation", from "order execution" to "order execution" and from "order release" to "maintenance order release". Furthermore, for model 2 on the left hand side, the functions "material planning", "order permit" and "order printing" were not mapped to any of the functions

Table 1: The seven EPCs containing the function "order execution" in the SAP reference model.

1. plant maintenance / breakdown maintenance processing / order / order,
2. plant maintenance / planned maintenance processing / order / order,
3. plant maintenance / project based maintenance processing / order / order,
4. plant maintenance / refurbishment processing in plant maintenance / order / order,
5. quality management / test equipment management / maintenance order / maintenance order,
6. quality management / test equipment management / service order / service order,
7. treasury / forex spot forward and swap transactions / transaction processing / transaction processing.

in the other model. Similarly, the functions "overall completion confirmation", "order settlement" and "order completion" in model 5 on the right hand side were not mapped to any of the functions in model 2.

The results presented in Table 2 give us valuable insights into the SAP reference model. First, they show that models containing the function "order execution" within the same module (i.e. "plant maintenance", "quality management" and "treaury") are exactly the same (in terms of behaviour). The question arises whether this is desirable, i.e. should there be so many copies of exactly the same model?

Furthermore, Table 2 also shows that the similarity between models in different modules is rather low. This implies that although these models contain a reference to exactly the same function "order execution", the contexts in which this function is executed are rather different and hence that the SAP reference model basically contains three different functions, which are all labelled "order execution".

Finally, it should be noted that the mere fact that two EPCs are 26.4% similar in itself is not too informative. However, in a querying scenario this measure can be used to rank several process models of the result set (for example, when looking for a model that is similar to model 1 of Table 1 the models 2,3 and 4 are more likely candidates than the

Table 2: Similarity between EPCs containing the function "order execution" in the SAP reference model.

	1	2	3	4	5	6	7
1	100 %	100 %	100 %	100 %	26.4 %	26.4 %	5.28 %
2		100 %	100 %	100 %	26.4 %	26.4 %	5.28%
3			100 %	100 %	26.4 %	26.4 %	5.28%
4				100 %	26.4 %	26.4 %	5.28%
5					100 %	100 %	5.01%
6						100 %	5.01%
7							100 %

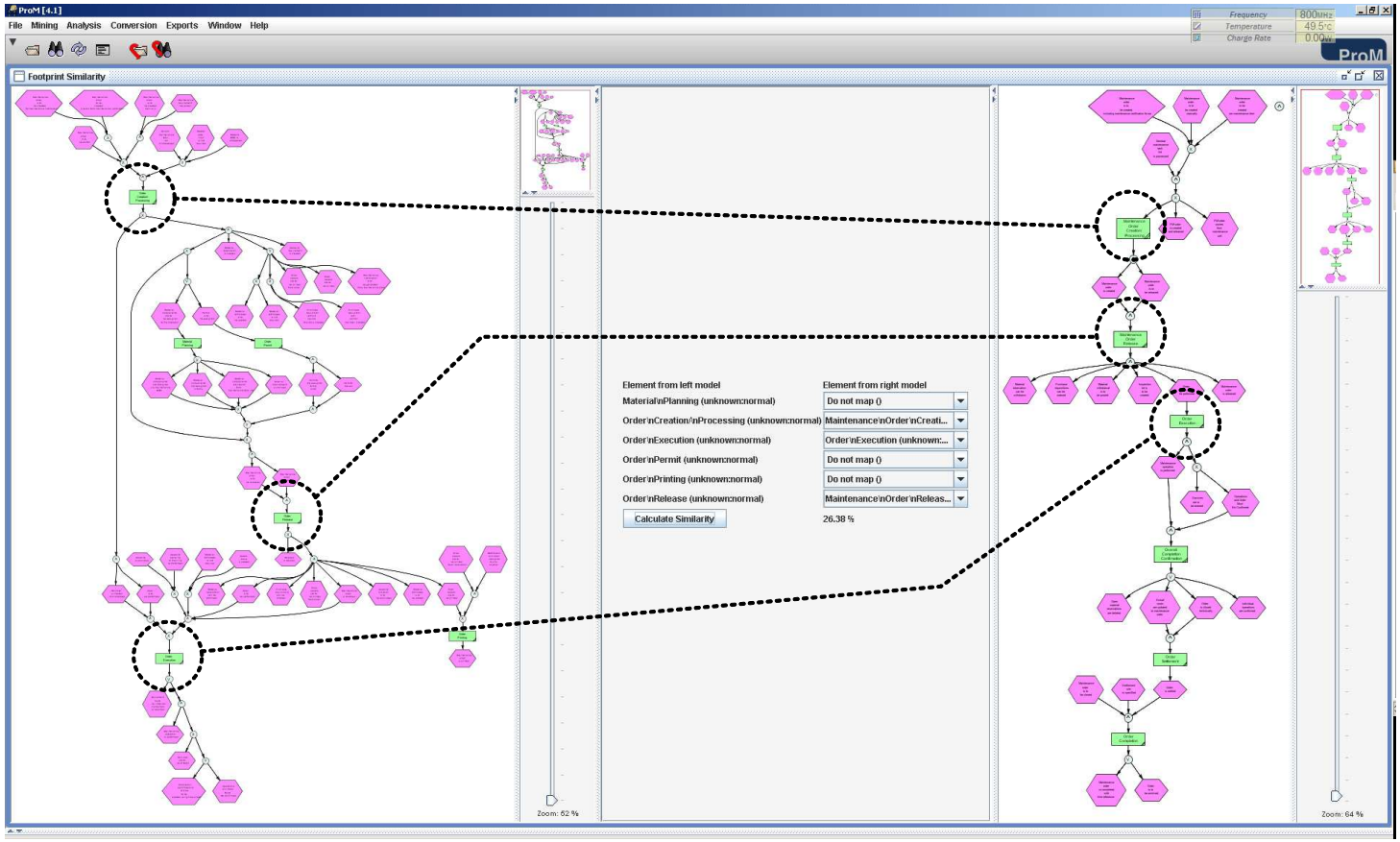

Figure 5: Comparison of two SAP models in ProM: $sim(G_2, G_5) = 0.2638$.

models 5,6 and 7) and in an merger scenario this value might be high enough to advocate an integration of the process models as is the case with models 1, 2, 3 and 4 and models 5 and 6.

5 Tool support

The (Pro)cess (M)ining framework *ProM* has been developed as a completely plug-able environment for process mining and related topics in the context of business process analysis. It can be extended by simply adding plug-ins, i.e., there is no need to know or to recompile the source code. Currently, more than 150 plug-ins have been added [DAV+05]. ProM is open-source and can be downloaded from `http://www.processmining.org`.

In the context of this paper, we developed an analysis plug-in for the comparison of causal footprints. This analysis plug-in, takes two causal footprints (or models that can be converted to causal footprints, e.g. Petri nets and EPCs, using the already available plug-ins presented in [DMA06]) and calculates their similarity. The plug-in requires the user to first map the elements of one model onto the elements of the other model. To support this step the plug-in suggests a mapping based on the labels of corresponding elements. Furthermore, by default, the analysis plug-in considers only functions for EPCs and only visible transitions for Petri nets, by projecting the footprints on these elements before calculating similarity. Figure 5 shows the comparison of two EPCs in ProM.

For the practical application of our results it is important to note that the ProM framework is currently capable of reading and writing EPCs to AML (native to the Aris Toolset), EPML (a standard for storing EPCs), VDX (the XML format of Visio), and the Aris graph format used by Aris PPM. For Petri nets, multiple formats are available, including the standard PNML format. ProM also supports other languages such as YAWL and BPEL. Since causal footprints are language independent, different models can be compared, e.g., the similarity of a Petri net and an EPC can be calculated.

6 Related Work

The concept of a causal footprint based similarity measure relates to two research areas: (1) declarative approaches for process specification and (2) similarity of process models.

Causal graphs (or causal footprints) can be seen as a *declarative language*, i.e., instead of using explicit control-flow operators like sequence, iterations, etc., constraints are given. In this paper we consider two types of constraints (look-back links and look-ahead links) as introduced in [DMA06]. These can easily be translated into a temporal logic [MP91]. In [AP06] the Declarative Service Flow Language (DecSerFlow) is defined which includes the two types of constraints used in this paper but also many others. This illustrates that the approach presented in this paper could be extended to include other types of constraints in the causal footprint. In [AP06] it is shown how these can be represented in Linear

Temporal Logic (LTL) and a respective implementation of an LTL checker for event logs is included in the ProM framework [ABD05].

Existing work in the context of determining *similarity* between process models can be assigned to three categories: verification, integration, and degree of 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 [AAW06]. A general problem of such verification approaches is that it provides a true-false answer to the question whether two models are similar. The quantification of a degree of behavioral similarity between process models is an important contribution for the area of process model *integration*. While there are several approaches reported regarding *how* two models are integrated (see e.g. [PCS01, GRSS05, MS06]) the similarity degree gives an answer to the question *when* two process models might be a candidate for integration, e.g. in a merger situation.

Up to now, there is hardly any work reported on measuring the degree of *behavioral similarity* of two process models. In [AAW06] three similarity notions are discussed related to the structure, the state space, and the observed behavior of the models. The *structural similarity* counts connections between tasks that appear in both models and relates it either to the number of connections in the first benchmark model (called recall metric) or to those of the second model (called precision metric). Yet, it does not consider transitive order relations of tasks. The *naive behavioral similarity* is based on firing sequences of both models, but bears several problems with respect to termination, moment of choice, or finiteness of the state spaces. Instead, the authors propose a *behavioral similarity* metric based on the fitness of a set of event logs. This is indeed valuable for process mining, but not directly applicable for model comparison where no logs are present, or logs cannot be directly mapped to the higher-level description of the process (i.e. if there are transaction logs on one side, and a conceptual high level model on the other). In contrast to this work, causal footprint provide a rich set of information for comparison without the need to calculate the state space.

Finally, there are some approaches discussed for calculating the similarity of process models based on metadata (see e.g. [KB04, MS04, EKO07]). In contrast to the degree of similarity based on causal footprints, these contributions hardly use information about the behavior of the process models. Nevertheless, it seems possible to combine some of the ideas presented in these paper with causal footprints, e.g., semantical mappings between the labels of activities in different models.

7 Conclusion and Future work

In this paper, we presented a metric for comparing process models in different formats. We capture the *intent* of a process model by deriving a *causal footprint*. Such a footprint should be seen as an abstraction of the process behavior. As a causal footprint does

not require an executable semantics of the process modeling language, we can apply our analysis technique both to formal languages such as Petri nets and to conceptual/informal languages such as EPCs. Even though we use only these two languages throughout the paper to demonstrate the applicability of our technique, it can be easily adapted to other languages such as UML activity diagrams, BPMN, or BPEL. This is especially helpful regarding the heterogeneity of business process modeling languages. Furthermore, the causal footprints provide a rich set of information over models that can be utilized to determine the degree of similarity between them.

The degree of behavioral similarity based on causal footprints provides the basis for a more efficient querying and management of large collections of business process models, even if they are represented in different modeling languages. In particular the reengineering of such model collections would profit from our concepts since a similarity matrix might clearly indicate which pairs of process models are likely candidates for integration or deletion.

References

[Aal98] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[AAW06] W.M.P. van der Aalst, A.K. Alves de Medeiros, and A.J.M.M. Weijters. Process Equivalence: Comparing Two Process Models Based on Observed Behavior. In S. Dustdar, J.L. Faideiro, and A. Sheth, editors, *International Conference on Business Process Management (BPM 2006)*, volume 4102 of *Lecture Notes in Computer Science*, pages 129–144. Springer-Verlag, Berlin, 2006.

[ABD05] W.M.P. van der Aalst, H.T. de Beer, and B.F. van Dongen. Process Mining and Verification of Properties: An Approach based on Temporal Logic. In R. Meersman and Z. Tari et al., editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005*, volume 3760 of *Lecture Notes in Computer Science*, pages 130–147. Springer-Verlag, Berlin, 2005.

[ADK02] W.M.P. van der Aalst, J. Desel, and E. Kindler. On the Semantics of EPCs: A Vicious Circle. In M. Nüttgens and F.J. Rump, editors, *Proceedings of the EPK 2002: Business Process Management using EPCs*, pages 71–80, Trier, Germany, November 2002. Gesellschaft für Informatik, Bonn.

[ADMV06] W.M.P. van der Aalst, B.F. van Dongen, J. Mendling, and H.M.W. Verbeek. Fouten in SAP-referentiemodel (in Dutch). Automatisering Gids, 19th May 2006.

[AHKB03] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[AP06] W.M.P. van der Aalst and M. Pesic. DecSerFlow: Towards a Truly Declarative Service Flow Language. In M. Bravetti, M. Nunez, and G. Zavattaro, editors, *International Conference on Web Services and Formal Methods (WS-FM 2006)*, volume 4184, pages 1–23, 2006.

[BYRN99] R.A. Baeza-Yates and B.A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[CKL97] T. Curran, G. Keller, and A. Ladd. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Enterprise Resource Planning Series. Prentice Hall PTR, Upper Saddle River, 1997.

[DAV+05] B.F. van Dongen, A.K. Alves de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The ProM framework: A new era in process mining tool support. In *Application and Theory of Petri Nets 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 444–454. Springer-Verlag, Berlin, 2005.

[DE95] J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.

[DJVVA] B.F. van Dongen, M.H. Jansen-Vullers, H.M.W. Verbeek, and W.M.P. van der Aalst. Methods for EPC Verification and application to the Aris for MySAP reference models. *Computers in Industry (accepted for publication)*.

[DMA06] B.F. van Dongen, J. Mendling, and W.M.P. van der Aalst. Structural Patterns for Soundness of Business Process Models. In *EDOC '06: Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06)*, pages 116–128, Washington, DC, USA, 2006. IEEE Computer Society.

[EJL+99] H. Eertink, W. Janssen, P. Oude Luttighuis, W. B. Teeuw, and C. A. Vissers. A business process design language. In *FM '99: Proceedings of the Wold Congress on Formal Methods in the Development of Computing Systems-Volume I*, pages 76–95, London, UK, 1999. Springer-Verlag.

[EKO07] M. Ehrig, A. Koschmider, and A. Oberweis. Measuring similarity between semantic business process models. In J.F. Roddick and A. Hinze, editors, *Conceptual Modelling 2007, Proceedings of the Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM 2007)*, volume 67, pages 71–80, Ballarat, Victoria, Australia, 2007. Australian Computer Science Communications.

[GRSS05] G. Grossmann, Y. Ren, M. Schrefl, and M. Stumptner. Behavior based integration of composite business processes. In W.M.P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Business Process Management, 3rd International Conference, BPM 2005, Nancy, France, September 5-8, 2005, Proceedings*, volume 3649 of *Lecture Notes in Computer Science*, pages 186–204. Springer, 2005.

[KB04] M. Klein and A. Bernstein. Toward high-precision service retrieval. *IEEE Internet Computing*, 8(1):30–36, 2004.

[Kin06] E. Kindler. On the Semantics of EPCs: A Framework for Resolving the Vicious Circle. *Data and Knowledge Engineering*, 56(1):23–40, 2006.

[KNS92] G. Keller, M. Nüttgens, and A.W. Scheer. Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrücken, 1992.

[KT98] G. Keller and T. Teufel. *SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping*. Addison-Wesley, 1998.

[MADV06a] J. Mendling, W.M.P. van der Aalst, B.F. van Dongen, and H.M.W. Verbeek. Errors in the SAP Reference Model. BPTrends, June 2006.

[MADV06b] J. Mendling, W.M.P. van der Aalst, B.F. van Dongen, and H.M.W. Verbeek. Referenzmodell: Sand im Getriebe - Webfehler. *iX - Magazin für Professionelle Informationstechnik. (in German)*, pages 131–133, August 2006.

[MP91] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, New York, 1991.

[MS04] 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, *Advances in Databases and Information Systems, 8th East European Conference, ADBIS 2004, Budapest, Hungary, September 22-25, 2004, Proceesing*, volume 3255 of *Lecture Notes in Computer Science*, pages 306–321. Springer, 2004.

[MS06] 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.

[Mur89] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[MVD+07] J. Mendling, H.M.W. Verbeek, B.F. van Dongen, W.M.P. van der Aalst, and G. Neumann. Detection and Prediction of Errors in EPCs of the SAP Reference Model. *Data and Knowledge Engineering (accepted)*, 2007.

[PCS01] G. Preuner, S. Conrad, and M. Schrefl. View integration of behavior in object-oriented databases. *Data & Knowledge Engineering*, 36(2):153–183, 2001.

[RR98] W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.

[SWY75] G. Salton, A. Wong, and C.S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620, 1975.