Computers in Industry 63 (2012) 148–167

Contents lists available at SciVerse ScienceDirect

Computers in Industry

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

# A comparative survey of business process similarity measures

Michael Becker [a,*], Ralf Laue [b]

[a] Department of Business Information Systems, University of Leipzig, Germany
[b] University of Applied Sciences of Zwickau, Germany

ARTICLE INFO

*Article history:*
Received 26 November 2011
Accepted 29 November 2011
Available online 17 January 2012

*Keywords:*
Survey
Business process model
Similarity measures

ABSTRACT

Similarity measures for business process models have been suggested for different purposes such as measuring compliance between reference and actual models, searching for related models in a repository, or locating services that adhere to a specification given by a process model. The aim of our article is to provide a comprehensive survey on techniques to define and calculate such similarity measures.

As the measures differ in many aspects, it is an interesting question how different measures rank "similarity" within the same set of models. We investigated, how different kinds of changes in a model influence the values of different similarity measures that have been published in academic literature.

Furthermore, we identified eight properties that a similarity measure should have from a theoretical point of view and analysed how these properties are fulfilled by the different measures. Our results show that there are remarkable differences among existing measures.

We give some recommendations which type of measure is useful for which kind of application.



## 1. Introduction

Business process models, or just process models (PMs), are nowadays a common approach to analyse existing business processes and to create new processes in a structured way. They are used for purposes like supporting communication in organisations, documentation in projects, and training of employees [1]. This wide area of application has led to the existence of a tremendous amount of PMs. Large scale enterprises often own process repositories consisting of hundreds or even thousands of models [2], usually developed by different persons. A variety of techniques to manage these repositories are conceivable. They range from intelligent process repositories [3] to similarity search over the models.

So far, several approaches that follow the latter idea have been proposed. They aim to find PMs in a PM repository that are similar to a given query model. For this purpose, there is a need of a similarity measure that quantifies the similarity between models.

The goal of our article is to provide a comprehensive survey on techniques to define and calculate similarity measures between PMs. Furthermore, we will study the question how the different measures rank "similarity" within the same set of PMs. In our study, we investigated how different kinds of changes in a PM influence the values of different similarity measures that have been published in academic literature.

The general method for comparing two PMs consists of two steps: First, activity nodes in one model that correspond to activity nodes in the other model must be identified. In particular, if the models have been created in different organisations or if they describe a business process on different levels of detail, this can become a non-trivial task. This first step is, however, not in the focus of our article. We assume that a mapping between corresponding activity nodes in the process models to compare has been established, either by using one of the existing algorithms or based on experts' judgment. The interested reader can find a discussion of different mapping techniques in [4–6]. More general techniques that map model fragments instead of single nodes to each other are discussed in [7]. Once a mapping between the activities has been established, measures of similarity between the models can be computed in step two.

After explaining some basic concepts and symbols in Section 2, we will discuss desirable properties that a similarity measure should have in Section 3. The methods used in our literature survey and the applications for similarity measures that have been suggested in the literature are presented in Section 4. To facilitate comparability of the analysed measures, we calculate similarity between an example model and its variations which we present in Section 5. Following, in Section 6, we discuss the measures we have found in the literature. For each measure, we analyse which of the properties discussed in Section 3 are fulfilled, and we compute the similarity between our example model and its variants. From

* Corresponding author.
*E-mail addresses:* michael.becker@uni-leipzig.de (M. Becker), ralf.laue@fh-zwickau.de (R. Laue).

0166-3615/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.compind.2011.11.003

the observations made for the different measures, we give some recommendations which kind of measure is more or less useful for which purpose in Section 7. Finally, Section 8 concludes the paper.

## 2. Preliminaries

### 2.1. Business process models as graphs

Throughout this article, let $M_0$ be the model that should be compared with a set of other models. We will refer to these models as $M_1$ (when discussing the comparison of $M_0$ with a single model) and $M_2, \ldots$ (when discussing the comparison of $M_0$ with more than one model).

Each such model can be described as a directed graph $(N, E)$ with the set $N$ of nodes and the set $E$ of edges. $A \subseteq N$ is the subset of nodes that have a textual label assigned. Depending on the modelling language, this set can include either activities (i.e. the tasks that have to be executed in a business process) or activities and events. Without loss of generality, in this article we will refer to the members of $A$ as "activities" unless a separate discussion of events is necessary. We will use the notation $(N_i, E_i)$ (where $i = 0, 1, \ldots$) for describing a model $M_i$ as a graph. The set of activities of $(N_i, E_i)$ is denoted by $A_i$.

Other than activity and event nodes, process models can also include connectors (called "gateways" in the language BPMN) which are used to model parallel executions or choices among paths.

Fig. 1 shows two example process models $PM_0$ (above) and $PM_1$, modelled in the Business Process Modeling Notation [8]. In this notation, start and end events are depicted by circles, activities by rounded rectangles and "choice nodes" (a gateway that allows to chose exactly one of the outgoing arcs for further processing) by the symbol $\diamond$. All those elements belong to the set of nodes of a model.

The example model $PM_0$ of Fig. 1 can be expressed as a graph as follows:

$$PM_0 = (N_0, E_0) = (\{m_0, m_1, m_2, m_3, m_4, m_5, m_6\}, \{(m_0, m_1), (m_1, m_2), (m_2, m_3), (m_3, m_4), (m_4, m_2), (m_4, m_5), (m_5, m_6)\})$$

Furthermore, we define the set $\bullet n$ of the preceding nodes for a node $n \in N$ as

$\bullet n = \{m \in N \text{ such that } (m, n) \in E\}$ and analogously the set $n\bullet$ of the succeeding nodes of a node $n \in N$ as

$n \bullet = \{m \in N \text{ such that } (n, m) \in E\}$.

### 2.2. Trace, set of traces

A process model $(N, E)$ describes the temporal relations between possible executions of the activities $A \subseteq N$. For example, if $A \times A \ni (a_I, a_{II}) \in E$, this means that the execution of activity $a_I$ is followed by the execution of activity $a_{II}$. A *trace* (also called firing sequence in the domain of Petri nets) of $(N, E)$ is a finite or infinite sequence of activities from the set $A$, denoting the order in which the execution of activities from $A$ starts in an instance of the process. For example, the trace $\langle a_1, a_2, a_1 \rangle$ means that the process is instantiated by starting activity $a_1$. At a later point of time, $a_2$ starts, and even later the execution of $a_1$ is started for a second time. The length of a trace $\sigma$ is the number of its elements (or $\infty$) and will be denoted by $len(\sigma)$. We will use the symbol $\Sigma(M)$ to denote the set of all possible traces of a PM $M$.

In the upper example model $PM_0$ of Fig. 1, possible traces are: $\langle m_0, m_1, m_2, m_3, m_4, m_5, m_6 \rangle$ as well as $\langle m_0, m_1, m_2, m_3, m_4, m_2, m_3, m_4, m_5, m_6 \rangle$ ($m_3$ is repeated once), $\langle m_0, m_1, m_2, m_3, m_4, m_2, m_3, m_4, m_2, m_3, m_4, m_5, m_6 \rangle$ ($m_3$ is repeated twice), etc. As the activity $m_3$ can be repeated arbitrary often, $\Sigma(PM_0)$ is an infinite set. The set of traces of the lower model $PM_1$ of Fig. 1 contains exactly one trace, $\langle n_0, n_1, n_2, n_3, n_4, n_5 \rangle$.

Traces of process models can be represented as a string, i.e. a sequence of symbols. Based on this representation, it is possible to calculate the *longest common subsequence*. The longest common subsequence of two strings is a subsequence of both strings that contains the maximum number of symbols (preserving the symbol order). For example for the strings "123456" and "1x2y3z", the longest common subsequence is "123". A more formal definition and algorithms for calculating longest common subsequences can be found in [9].

We denote the length of the longest common subsequence of traces $\sigma_1$ and $\sigma_2$ as $len(lcs(\sigma_1, \sigma_2))$.

### 2.3. Mapping function

In most approaches, the algorithm for calculating similarity measures starts with establishing a mapping between the nodes in $M_0$ and $M_1$. Such a mapping describes which activity in $M_1$ "corresponds" to an activity in $M_0$. Formally, a mapping is described by a partial function that assigns nodes of $M_0 = (N_0, E_0)$ to the "corresponding" nodes of $M_1 = (N_1, E_1)$. Throughout this article, we will denote this mapping function with $map : N_0 \to N_1$. In many approaches, only the activities are mapped. In these cases, we have a mapping function $map : A_0 \to A_1$. While some approaches require $map$ to be injective, others do not.

For the models shown in Fig. 1, a reasonably defined mapping function $map : PM_0 \to PM_1$ could be: $map(m_0) = n_0$, $map(m_1) = n_1$, $map(m_3) = n_2$, $map(m_5) = n_3$, $map(m_6) = n_5$. In this case, $map$ is neither total nor injective.

$map$ is either defined by human judgment or automatically by making use of a function $corr : N_0 \times N_1 \to [0, 1]$ that quantifies the correspondence (or similarity) between single nodes. The basic idea is to define $map$ such that the values of $corr(n, map(n))$ are close to 1 for many nodes $n$.

We regard two models $M_0$ and $M_1$ as equivalent to each other (symbol: $M_0 \equiv M_1$), iff $map$ is a graph isomorphism of $M_0$ and $M_1$, i.e. $map$ is bijective and $(n, m) \in E_0 \Leftrightarrow (map(n), map(m)) \in E_1$.

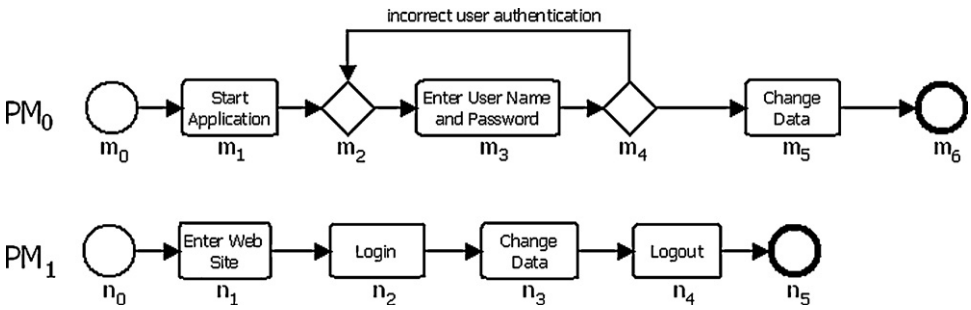



**Fig. 1.** Two simple example models.

### 2.4. Distance and similarity measures, further symbols

Let **M** be the set of process models. A *distance measure dist* is a function $dist : \mathbf{M} \times \mathbf{M} \rightarrow \mathbb{R}^+ \cup \{0\}$.

A *similarity measure* is a function $sim : \mathbf{M} \times \mathbf{M} \rightarrow [0, 1]$. The formula

$$sim(x, y) = \frac{1}{1 + dist(x, y)} \quad (1)$$

can be used for transforming a distance measure into a similarity measure or vice versa.

In Section 6, we will discuss several alternatives to define the functions *map*, *corr*, *dist* and *sim*, i.e. in each subsection those functions will be defined differently.

Throughout this article, the symbol # *K* will be used to denote the number of elements of a set K.

## 3. Desirable properties of distance and similarity measures

Santini and Jain [10] point out that a number of dissimilarity measures proposed in the literature assume that those measures are distance measures in a metric space. (**M**, *dist*), the set of all process models **M** with a distance measure *dist*, becomes a metric space, if the following properties hold:

**Property 1** $dist(M_0, M_1) \geq 0 \quad \forall M_0, M_1 \in \mathbf{M}$ (non-negativity)
**Property 2** $dist(M_0, M_1) = dist(M_1, M_0) \quad \forall M_0, M_1 \in \mathbf{M}$ (symmetry)
**Property 3** $dist(M_0, M_1) = 0 \Leftrightarrow M_0 \equiv M_1$
**Property 4** $dist(M_0, M_2) \leq dist(M_0, M_1) + dist(M_1, M_2)$ (triangle inequality)

For measuring the "dissimilarity" distance between PMs, it is reasonable to require Property 1 and Property 2. Property 3 that says that the distance between two models is 0 only if the models are identical is too strict for certain application areas. The same set of traces $\Sigma(M)$ (i.e. the same set of possible executions of activities of a model *M*) can be modelled in different ways. For example, the model shown in Fig. 3(a) (see Section 5) has the same set of traces as the model shown in Fig. 3(b). A distance measure that calculates the distance between both models as 0 would correctly describe the fact that both models show exactly the same business process. A more relaxed requirement is that $dist(M_0, M_1)$ is 0, iff both models have the same set of traces.

For our purposes, the set of traces $\Sigma(M_0)$ and $\Sigma(M_1)$ are considered as being the same (symbol: $\Sigma(M_0) \equiv \Sigma(M_1)$) if $\langle s_1, s_2, \dots \rangle \in \Sigma(M_0)$ implies that $\langle map(s_1), map(s_2), \dots \rangle \in \Sigma(M_1)$ and vice versa, $\langle t_1, t_2, \dots \rangle \in \Sigma(M_1)$ implies that there is a $\langle s_1, s_2, \dots \rangle \in \Sigma(M_0)$ such that $map(s_i) = t_i \; \forall \, i$.

With this interpretation of equality between sets of traces, Property 3 can be substituted by the less strict requirement:

**Property 3a:**
$dist(M_0, M_1) = 0 \Leftrightarrow \Sigma(M_0) \equiv \Sigma(M_1)$.

Property 4, the triangle inequality, is not essential for measuring the dissimilarity (distance) between PMs (or for (dis)similarity measures in general, see [11]). Therefore, we will not examine the suggested measures with respect to this property. It is a useful property anyway, because a distance measure that fulfils all four properties given above allows to organise a PM repository using data structures in which the search for similar models is very fast [12].

From an information-theoretic discussion of the concept of similarity (see [11,13]), one more requirement for a similarity measure can be derived: Such a measure should take into consideration both the commonality between two models and their differences (**Property 5**). For example, we would not get a good similarity measure by just counting the number of activities that are shared among two models without relating this number to the overall number of activities in the models: If two models with 20 nodes have 15 node names in common, it would be reasonable to say that they are more similar to each other than two models with 200 nodes from which 15 node names can be found in both models.

As mentioned before, the definition of the function $map : A_0 \rightarrow A_1$ is outside the main focus of this article. We just assume that such a mapping has been established. The approaches that calculate *map* automatically start with a function $corr : A_0 \times A_1 \rightarrow [0, 1]$ which quantifies the similarity between activities. It would be a desirable property of a similarity measure $sim : \mathbf{M} \times \mathbf{M} \rightarrow [0, 1]$ if the information gained from the similarity measure $corr : A_0 \times A_1 \rightarrow [0, 1]$ between *activities* would be considered in the calculation of the similarity measure *sim* between the *models as a whole* (**Property 6**). This is illustrated in Fig. 2, showing three sequential models $M_0$, $M_1$ and $M_2$ with four activities and the mappings between them (as dotted arrows). Assume that

$$1 = corr(\text{"confirm draft"}, \text{"confirm draft"})$$
$$> corr(\text{"confirm draft"}, \text{"dismiss draft"})$$

and that

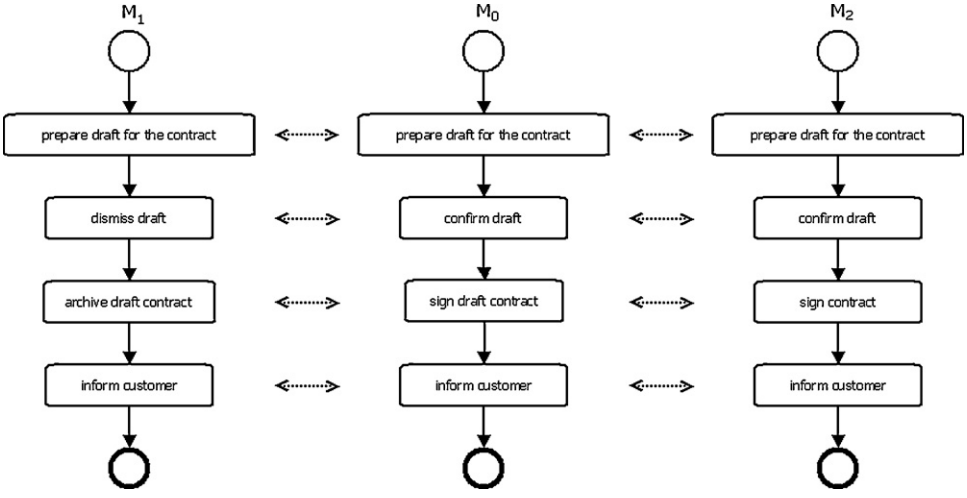



**Fig. 2.** Models with corresponding labels; the labels indicate that $M_2$ should be regarded as more similar to $M_0$ than $M_1$.

**Table 1**
Summary of the properties for distance measures defined in Section 3.

| | |
|---|---|
| 1 | Distance measure is non-negative |
| 2 | Distance measure is symmetric |
| 3 | Distance measure is 0 only if the models are the same (up to mapping by *map*) |
| 3a | Distance measure is 0 only if the models have the same set of traces (up to mapping by *map*) |
| 4 | Distance measure fulfils the triangle inequality |
| 5 | Distance measure considers both commonalities and differences |
| 6 | Distance measure takes similarity measure between activities into account |
| 7 | Distance measure is defined for arbitrary process models |
| 8 | Distance measure can be computed efficiently |

$$1 > corr(\text{“sign draft contract“}, \text{“sign contract“})$$
$$= corr(\text{“sign draft contract“}, \text{“archive draft contract“})$$

(which could be the result of *corr* defined as a simple word-by-word comparison). In such a case, it would be desirable that the result that the activities in $M_2$ are more similar to the activities in $M_0$ than those in $M_1$ would not “get lost” when the similarity measure *sim* is calculated, i.e. we would prefer to have $sim(M_0, M_1) < sim(M_0, M_2)$ instead of $sim(M_0, M_1) = sim(M_0, M_2)$.

Furthermore, it is reasonable to require that a distance or similarity measure can be applied for comparing arbitrary PMs without imposing additional syntax restrictions (such as that the model must not contain loops) (**Property 7**). And last but not least, there is another requirement that is related to the computational complexity of the calculation of *dist* or *sim*. In simple terms, it should be possible to calculate the values of distance / similarity measures efficiently (**Property 8**). Approaches that require the calculation of the whole set of traces of a PM often would not fulfil this requirement.

Table 1 gives a short overview of the properties defined in this section.

## 4. Literature research

### 4.1. Methods of the literature research

The findings we present in this article are based on an extensive literature review conducted between March 2010 and May 2011. According to the taxonomy given in [14], our review can be classified as follows:

- Scope: state-of-the-art presentation concerning approaches to calculate PM similarity
- Focus: comprehend research methods and technologies
- Goal: summarise findings
- Organisation: conceptual using the different types of similarity measures established in Section 6.1 to 6.5
- Perspective: neutral
- Audience: specialised scholars can identify similarities and differences between existing approaches to calculate PM similarity
- Coverage: exhaustive

The central starting points for our survey were the digital libraries of ACM, IEEE, Springer, and Elsevier (SciVerse ScienceDirect). We have searched for approaches to calculate similarity between process models using the search terms “process model” and “similarity”. Based on titles and abstracts and, if necessary, by reviewing the full paper, we identified 25 academic works published between 2004 and 2011 as relevant for our survey. This way, we received all papers but [12] which was only available at the authors' personal website.

**Table 2**
Analysed works and their application area.

| Measure | Simplify changes | Merge processes | Facilitate reuse | Manage repositories | Automate execution | Assure compliance | Discover services |
|---|---|---|---|---|---|---|---|
| 6.1.1 | × | × | × | | | | |
| 6.1.2 | | | × | × | × | | |
| 6.1.3 | | × | | | | | |
| 6.1.4 | × | | | | | | |
| 6.1.5a | | | | × | | | |
| 6.1.5b | × | | | | | | |

| Measure | Simplify changes | Merge processes | Facilitate reuse | Manage repositories | Automate execution | Assure compliance | Discover services |
|---|---|---|---|---|---|---|---|
| 6.2.1a | | | × | × | × | | |
| 6.2.1b | | | | | | | × |
| 6.2.2 | | × | | | | | |
| 6.2.3 | | | | × | | | |
| 6.2.4 | × | | | | | × | |
| 6.2.5 | × | | | | | | |
| 6.2.6 | × | × | | | | × | |

| Measure | Simplify changes | Merge processes | Facilitate reuse | Manage repositories | Automate execution | Assure compliance | Discover services |
|---|---|---|---|---|---|---|---|
| 6.3.1a | × | | | | | | |
| 6.3.1b | × | | | | | | |
| 6.3.2 | | × | | | | | |
| 6.3.3 | | × | | | | | |
| 6.3.4 | | | × | × | × | | |
| 6.3.5 | | | | | | | × |

| Measure | Simplify changes | Merge processes | Facilitate reuse | Manage repositories | Automate execution | Assure compliance | Discover services |
|---|---|---|---|---|---|---|---|
| 6.4.1 | | | | | | × | |
| 6.4.2 | | | | × | | × | |
| 6.4.3 | | | | | | × | × |

### 4.2. Application spectrum for similarity measures

Calculating similarity between PMs is a task performed in a wide range of applications concerning business process management. In the literature we have surveyed, several applications for similarity measures have been suggested. They can be grouped into seven categories: "simplify changes", "merge processes", "facilitate reuse", "manage PM repositories", "automate process execution", "assure compliance with normative models", and "discover services". It should be noted that these activities are not independent from each other. For example, simplifying changes is closely connected to managing process variants. In the following, each of these groups is presented in more detail, while Table 2 shows a summary of the application areas suggested by the various authors for their measures. Table 2 can serve as a first guide for studying the different measures which will be introduced in detail in Section 6.

#### 4.2.1. Simplify changes in process variants

Organisations can react on changing customer requirements by adapting existing processes. In doing so, various related models are established that are all more or less similar to each other. To simplify management and facilitate reuse of these process variants, it is necessary to establish a measure that captures their similarity. Therefore, it is possible to react on new user requirements by searching for process variants that satisfied similar requirements in the past.

#### 4.2.2. Merge processes

Merging PMs is a common activity executed in the case of company mergers and in collaborations beyond company borders. When business units of different organisations are consolidated, it can be assumed that process overlaps exist. For example, [15] reported of an organisation having several subsidiaries where every subsidiary managed its own ERP system resulting in more than 200,000 PMs. During integration it is necessary to integrate these systems and to identify process overlaps.

#### 4.2.3. Facilitate reuse

A cross-sectional goal that can be achieved by targeting various other application areas for similarity calculations is to facilitate reuse of PMs. Similar to reusing components in software engineering (see e.g. [16] for code reuse), reusing PMs promises to reduce time and costs. Therefore, it is necessary to find existing PMs and reuse them in the right context.

#### 4.2.4. Manage PM repositories

Due to the vast amount of existing PMs, organisations usually store these models in process repositories. These repositories provide various functions, such as adding and removing models, annotating models and searching for models [3]. Before new models can be added to a repository, it is useful to check whether similar or even identical models are already stored in the repository. Furthermore, repositories are useless without efficient querying (provided by similarity searches) and browsing facilities.

#### 4.2.5. Automate process execution

Automation is usually concerned in SOA applications. During execution, services may be called depending on user requirements established at runtime. Furthermore, existing services may fail, e.g. due to a computer failure. In this case, it may be necessary to find similar services that are able to provide the same or similar functionality.

#### 4.2.6. Assure compliance with normative models

Reference models are a common approach to improve the process of developing new PMs [17]. Based on a given reference process, application specific processes can be established. Reference models often contain necessary legal requirements for specific domains. Therefore, it is often necessary to measure the compliance degree between a given reference model and its application specific implementation.

#### 4.2.7. Discover services

Closely connected to the goal of automation is service discovery. In SOA applications, one common task is to search for

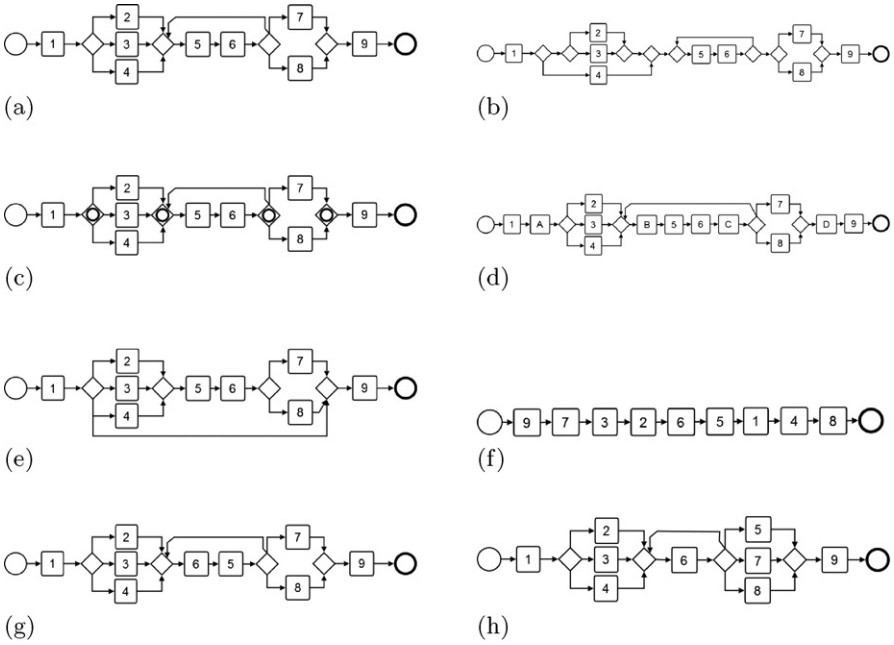



**Fig. 3.** Initial model $V_0$ and variants $V_1 \ldots V_7$. (a) $V_0$: Original BPMN model. (b) $V_1$: Model with same set of traces as $V_0$. (c) $V_2$: Model with modified connector types. (d) $V_3$: Model with additional activities. (e) $V_4$: Model with modified control flow. (f) $V_5$: Model with a modified control flow. (g) $V_6$: Model with a modified order of activities 5 and 6. (h) $V_7$: Model with activity 5 moved inside the second control block.

**Table 3**
Adherence to properties and similarity values of [22].

| Similarity score based on common activity names [22] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – no | 3a – no | 5 – yes | 6 – yes | 7 – yes | 8 – yes |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 1.00 | 0.82 | 1.00 | 1.00 | 1.00 | 1.00 |

services satisfying specific user requirements. If this task can be automated it is possible to call services dynamically and to make reuse of existing services possible.

## 5. Model changes

The approach followed in our survey is as follows: Starting from a moderately sized model $V_0$ (shown in Fig. 3(a)), we apply different change operations as described in [18–20], resulting in seven variants of the original PM. We selected the model variants such that the control-flow related change operations from [18–20] are represented. For the various similarity measures discussed in this survey, we compute the similarity between the original model $V_0$ (Fig. 3(a)) and each of its variants $V_1$, ..., $V_7$. If the original authors of a measure described it as a distance measure rather than a similarity measure, we use Eq. (1) for transforming the distance measure into a similarity measure.

The model variants are shown in Fig. 3 as PMs in the Business Process Modeling Notation [8]. The process $V_0$ starts with executing activity 1. After executing this activity, exactly one of the activities 2, 3 or 4 is performed (based on certain decision that is not explicitly described in the model). Afterwards, the activities 5 and 6 are executed one or more times, followed by exactly one of the activities 7 or 8. Finally, the process instance terminates after executing activity 9.

First, we modify the original model $V_0$ of Fig. 3(a) by splitting the XOR connectors into more than one connector (see Fig. 3(b)). Note that $\Sigma(V_1) = \Sigma(V_0)$. Next, we change the types of connectors: In model variant $V_2$ (Fig. 3(c)), all XOR connectors ($\diamond$) have been replaced by inclusive OR connectors ($\Diamond$). This means that in the blocks with the three activities 2, 3 and 4 and the two activities 7 and 8, *one or more* of the activities can be performed in parallel.

In variant $V_3$ (Fig. 3(d)), four additional activities A, B, C and D have been added to the original model.

Model variant $V_4$ (Fig. 3(e)) has exactly the same nodes as $V_0$, but one arc has been added while another one has been deleted. This means that in $V_4$, it is possible to skip all activities but 1 and 9, and there is no loop allowing activities 5 and 6 to be executed more than once. Variant $V_5$ (Fig. 3(f)) contains the same activities as $V_0$, but no connectors at all. The order of the activities does not correspond to the order in which the activities occur in traces of $V_0$. In model variant $V_6$ (Fig. 3(g)), the order of activities 5 and 6 has been changed. Finally, in model variant $V_7$, (Fig. 3(h)), activity 5 has been moved inside the second conditional control block.

## 6. Measures

In this section we present the analysed approaches for calculating similarity between PMs in detail. The approaches were identified based on the literature survey illustrated in Section 4.

In Section 6.1 we analyse approaches based on correspondences between the elements of a process model. Since process models can be represented as graphs as shown in Section 2.1, it is a reasonable idea to study the applicability of graph algorithms on similarity calculation. Approaches following this idea are presented in Section 6.2. Similarity measures considering the dependencies between activities in a PM, e.g. the order of their occurrence are shown in Section 6.3. Finally, we present approaches that are based on the set of traces (see Section 2.2) in Section 6.4.

Every presentation is enriched by a table containing information about the adherence to the properties in Section 3 and the absolute similarity values for the similarity between model $V_0$ and the models $V_1 \dots V_7$ from Section 5. Furthermore, we give a brief explanation of the parameters and (if necessary) adaptations used in our calculation of the similarity values and discuss each measure.

To enhance the reproducibility of our findings we developed a publicly available[1] application. It is based on the well-known ProM framework for process mining [21] and provides an extensible API. Currently, 15 of the presented measures are implemented, and we will add missing measures in the future. Using this application, it is possible to analyse the impact of various parameters when calculating similarity (e.g. size of models, amount of text in models). The source code contains detailed comments on the parameters and strategies for those measures whose original description allows some degree of freedom in the implementation.

### 6.1. Correspondence between nodes and edges in the PM

#### 6.1.1. Similarity score based on common activity names

Akkiraju and Ivan [22] measure similarity of process models solely based on the number of equally labelled activities, i.e. on the number of activities that occur in both models. The so called semantic similarity score between model $M_0$ with the set of activities $A_0$ and model $M_1$ with the set of activities $A_1$ is defined as $sim(M_0, M_1) = 2 \cdot (\#(A_0 \cap A_1)/(\#A_0 + \#A_1))$.

Any two of the example models $V_i$ (with the only exception $V_3$) have a similarity score of $2 \cdot (9/(9+9)) = 1$ irrespective of the calculation order.

*Calculation settings and results:* Table 3 shows the results of the similarity calculations between model $V_0$ and models $V_1 \dots V_7$ according to [22] as well as the adherence of this measure to the properties given in Section 3. The results were obtained by using the labels of activities as activity names.

*Discussion:* While this similarity measure is very simple and fast to compute, its major drawback is that changing the structure of processes (e.g. changing the order of activities or inserting connectors) does not influence the similarity measure in any way. Furthermore, its application is limited to a domain with controlled vocabulary (as the authors of [22] state, too). The approach fails both in multilingual and in inter-organisational environments due to different vocabularies. In Section 6.1.3 we discuss a similar approach avoids the restriction of activity labels to a controlled vocabulary.

#### 6.1.2. Label matching similarity

Dijkman et al. study several similarity measures in [4]. The first and simplest one is called label matching similarity. It builds on a function *corr* that calculates a label-based similarity score between nodes in $A_0$ and $A_1$. The mapping function $map : M_0 \to M_1$ is defined such that $\sum_{x \in A_0} corr(x, map(x))$ takes its maximum value. Optionally, a threshold can be used that disregards a similarity score if $corr(x, y)$ is smaller than a given value. This means that instead of *corr*, the function

[1] https://sourceforge.net/projects/prom-similarity/

**Table 4**
Adherence to properties and similarity values of [4].

| Label matching similarity [4] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – no | 3 – no | 3a – no | 5 – yes | 6 – yes | 7 – yes | 8 – yes |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 1.00 | 0.82 | 1.00 | 1.00 | 1.00 | 1.00 |

**Table 5**
Adherence to properties and similarity values of [6].

| Syntactic, linguistic and structural similarity of activity labels [6] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – no | 3 – no | 3a – no | 5 – yes | 6 – yes | 7 – yes | 8 – yes |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |

$$corr'(x,y) = \begin{cases} corr(x,y) & \text{if } corr(x,y) \geq threshold, \\ 0 & \text{otherwise} \end{cases}$$

is used.

The label matching similarity is defined as $sim(M_0, M_1) = \frac{2\sum_{x \in A_0} corr(x, map(x))}{\#A_0 + \#A_1}$.

*Calculation settings and results:* To establish correspondences between nodes, we use the Levenshtein string edit distance [23] as function *corr*. Furthermore, to match two activities, we use a threshold of 0.5. Since we only have single-letter words in our example models the resulting similarities between individual activities is either 0 or 1. The resulting similarity values are shown in Table 4.

*Discussion:* As can be seen from Table 4, the similarity values for the approach proposed by [4] match with the results established by [22] (Section 6.1.1) This is due to the fact that the measure of [4] also does not take into account any information about the order of nodes. For example, $sim(V_0, V_5)$ would be 1 while $sim(V_0, V_3)$ would be $\frac{18}{21} < 1$ only – a result that is counter-intuitive if we are interested in the similarity of the modelled behaviour.

### 6.1.3. Syntactic, linguistic and structural similarity of activity labels

Ehrig et al. [6] measure the similarity of PMs based on so-called semantic business process models - predicate transition Petri nets transformed in an ontological representation.

Let $A_0 = \{a_1^0, a_2^0, \ldots a_{\#A_0}^0\}$ and $A_1 = \{a_1^1, a_2^1, \ldots a_{\#A_1}^1\}$ be the sets of activities of the models. Ehrig et al. describe three measures for similarity between the label of an activity $a_0 \in A_0$ and the label of an activity $a_1 \in A_1$.

First, the Levenshtein string edit distance [23] is used to calculate the *syntactic similarity*. Second, *linguistic similarity* handles synonyms in element labels using WordNet [24] and takes the meaning of the words in the label into account. Finally, *structural similarity* compares the context (such as attribute names and values or succeeding nodes) of single activities. The similarity measure $corr(a_0, a_1)$ between activity $a_0 \in A_0$ and activity $a_1 \in A_1$ is aggregated by the weighted combination of the three similarity types.

The similarity of the models as a whole is calculated as

$$sim(M_0, M_1) = \frac{1}{\#A_0} \sum_{i=1}^{\#A_0} \max_{j=1,\ldots\#A_1} corr(a_i^0, a_j^1) \quad (2)$$

*Calculation settings and results:* Table 5 contains the similarity values for our variants. We defined *map* such that activities having the same label are mapped to each other. Since the activity labels of the variants shown in Fig. 3 are single-letter words, there is (with the exception of variant $V_3$) always a pair of activities with a syntactic similarity of 1. This results in the fact that linguistic and structural similarity is not taken into account. By using a non-injective mapping, $sim(V_0, V_3)$ would be higher because we could establish a mapping *map* such that $map(1) = (1, A)$, $map(5) = (B, 5)$, $map(6) = (6, C)$, and $map(9) = (D, 9)$.

*Discussion:* The approach of Ehrig et al. focuses to a great extent on the similarity of activity labels. Structural similarity is only taken into account if labels are not equal or if they are not synonyms of each other. Resulting from this, PMs with the same activity names will always have similarity 1 independently from any structural changes.

Following Eq. (2), the approach of Ehrig et al. maps the activities of $A_0$ to activities of $A_1$ and ignores activities in $A_1$ that are not contained in $A_0$. This results in a similarity of 1 when comparing $V_0$ with $V_3$ (activities *A*, *B*, *C*, and *D* are ignored). On the other hand, $sim(V_3, V_0)$ is not 1 but 9/13 because nine activities are matched with a similarity of 1 and the additional four activities are not matched and have a similarity of 0 (assuming a one-to-one-mapping). This means that Property 2 is violated – the measure is not symmetric. We believe the value of [6] lies more in a discussion of concepts to define the mapping function *map* between activities than in the definition of similarity measures between PMs as a whole.

### 6.1.4. Feature-based similarity estimation

Yan et al. [25] address the problem of searching a collection of PMs for models that are similar to a query model. They point out that it is inefficient to compare each model in the collection with the query model. As a solution, Yan et al. suggest to build computational efficient indices for quickly finding models that have many features in common with the query model. In this procedure, *features* are defined as activity labels as well as the position that a node has within the structure of the PM graph.

First, the Levenshtein string edit distance [23] is used for computing the similarity of activity labels. Second, the role of the nodes is taken into account which is defined for each node as shown in Table 6. For a node *x*, *roles*(*x*) is the set of all roles for *x*. It is possible that a node can have more roles at once, for example being a split as well as a join.

For two PM $M_0 = (N_0, E_0)$ and $M_1 = (N_1, E_1)$, *structural similarity* $corr_{str}$ between two nodes $n \in N_0$, $m \in N_1$ is calculated as follows:

$$corr_{str}(n,m) = \begin{cases} 1 & \text{if } N_0 = \{n\} \wedge N_1 = \{m\} \\ 1 - \frac{|\#n\bullet - \#m\bullet|}{2(\#n\bullet + \#m\bullet)} & \text{if start} \in \text{roles}(n) \cap \text{roles}(m) \wedge \\ & \text{if stop} \notin \text{roles}(n) \cap \text{roles}(m), \\ 1 - \frac{|\#\bullet n - \#\bullet m|}{2(\#\bullet n + \#\bullet m)} & \text{if stop} \in \text{roles}(n) \cap \text{roles}(m) \wedge \\ & \text{if start} \notin \text{roles}(n) \cap \text{roles}(m), \\ 1 - \frac{|\#n\bullet - \#m\bullet|}{2(\#n\bullet + \#m\bullet)} - \frac{|\#\bullet n - \#\bullet m|}{2(\#\bullet n + \#\bullet m)} & \text{otherwise.} \end{cases}$$

The similarity of nodes (both similarity of activity labels as structural similarity as defined by the above formula) is used for establishing the mapping function *map* that assigns nodes from $N_1$

**Table 6**
Roles for activity nodes in a PM.

| Role | Incoming edges | Outgoing edges |
|---|---|---|
| start | 0 | |
| stop | | 0 |
| regular | 1 | 1 |
| split | | ≥2 |
| join | ≥2 | |

to "similar" nodes in $N_0$. A mapping between $n \in N_0$ and $m \in N_1$ is established if and only if the values for label similarity and structural similarity between $n$ and $m$ are greater than some cutoff value.

Yu et al. derive the similarity measure *sim* for PMs from this function *map* by relating the number of nodes with a match to the overall number of nodes in the models to compare.

*Calculation settings and results:* Table 7 shows the results of the similarity calculations between model $V_0$ and the variants $V_1 \ldots V_7$. As proposed by [25], we use different threshold values to match individual nodes with each other. Two nodes are matched, if their labels are similar to a high degree (a threshold of 0.8) or if their roles are similar and their labels are similar to a medium degree (thresholds of 1.0 for role similarity and 0.2 for label similarity). In their work, Yan et al. identify so-called discriminative roles. A role is discriminative, if the number of nodes having this role is small enough compared to the overall number of nodes in a process model collection. However, we do not take discriminative power into account, since we only have a small number of process variants.

*Discussion:* As the routing effects of connectors (XOR, AND, OR) in a model are ignored, the same problems as discussed in Section 6.1.1 and 6.1.3 can be observed. However, the aim of Yan et al. is not to find "similar" models (with the meaning of "similar behaviour"), but rather "related" models. Given the fact that they present a fast implementation of their algorithm, the benefit of the approach is that it can be used as a first step to filter potentially relevant models from a large model collection. In a second step, it is possible to calculate similarity measures only for those models from the collection that have not been disregarded as irrelevant.

### 6.1.5. Percentage of common nodes and edges in the graph

Similar to the label matching similarity measures discussed in the previous subsections, Minor et al. [26] suggest a measure that relates the number of nodes and edges that can be found in both $M_0 = (N_0, E_0)$ and $M_1 = (N_1, E_1)$ to the overall number of nodes and edges in both models. As the purpose of the work by Minor et al. is to compare models that are adapted versions of a "template" model, it can be assumed that two nodes can be regarded as identical if and only if they have exactly the same label. This means that the Function *map* is simply the identity.

For a strictly sequential PM (i.e. one which has only activities, but no split or join nodes), the similarity measure is defined as:

$$sim(x,y) = 1 - \frac{\#(N_0 \backslash N_1) + \#(N_1 \backslash N_0) + \#(E_0 \backslash E_1) + \#(E_1 \backslash E_0)}{\#N_0 + \#N_1 + \#E_0 + \#E_1} \quad (3)$$

For the general case of a PM with split and join nodes, three methods for transforming the PM in another graph (called approximation graph) are suggested in [26]. Method 1 simply ignores the splits and joins. Method 2 adds the types of splits and joins that can be found along a path from one activity to another as attributes to the arcs between activities. Method 3 uses exactly one "artificial" node per type of split- or join node, regardless of the number of occurrences of this kind of split- or join node in the PM. Fig. 4 illustrates the three methods of transforming a PM into its approximation graph for our example model $V_0$. After transforming the original PM, Eq. (3) is used for comparing the approximation graphs.

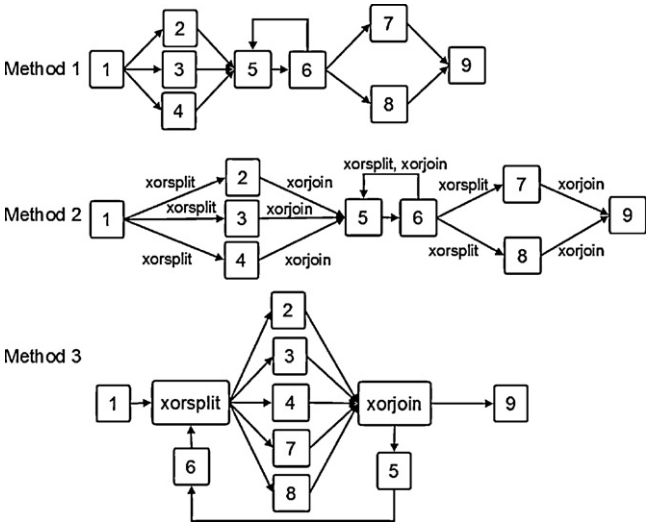



**Fig. 4.** Three methods to transform model $M_0$ from Fig. 3(a) into its approximation graph.

A similar approach is presented by Huang et al. in [27]. First, the function *corr* is calculated; Huang et al. do not suggest any specific *corr*-measure. Let $A_0 = \{a_1^0, a_2^0, \ldots a_{\#A_0}^0\}$ and $A_1 = \{a_1^1, a_2^1, \ldots a_{\#A_1}^1\}$ be the sets of activities of the models to compare and $E_0 = \{e_1^0, e_2^0, \ldots e_{\#E_0}^0\}$ and $E_1 = \{e_1^1, e_2^1, \ldots e_{\#E_1}^1\}$ the sets of its edges.

The overall similarity between the activity sets $A_0$ and $A_1$ is then defined as:

$$\frac{\sum_{i=1}^{\#A_0} \max_{j=1,\ldots,\#A_1}(corr(a_i^0, a_j^1)) + \sum_{j=1}^{\#A_1} \max_{i=1,\ldots,\#A_0}(corr(a_i^0, a_j^1))}{\#A_0 + \#A_1} \quad (4)$$

Second, the PMs to compare are transformed into a weighted graph representation similar to method 2 illustrated in Fig. 4. However, instead of labelling edges with the respective connector type, weights are assigned to edges. Outgoing edges resulting from XOR-splits are assigned with the weight of $\frac{1}{\#x_\bullet}$. This weight is added to edges until the next XOR-join. The transformation of $V_0$ from Fig. 3(a) into a weighted graph is shown in Fig. 5.

The similarity of two edges $e_I = (a_I, b_I)$ and $e_{II} = (a_{II}, b_{II})$ is based on the similarity of the linked activities and defined as:

$$sim_{ed}((a_{II}, b_{II}), (a_{II}, b_{II})) = \frac{corr(a_I, b_I) + corr(a_{II}, b_{II})}{2} \quad (5)$$

**Table 7**
Adherence to properties and similarity values of [25].

| Feature-based similarity estimation [25] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property … | 1 – yes | 2 – yes | 3 – no | 3a – no | 5 – yes | 6 – yes | 7 – yes | 8 – yes |
| Similarity between $V_0$ and … | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 1.00 | 0.82 | 1.00 | 1.00 | 1.00 | 1.00 |

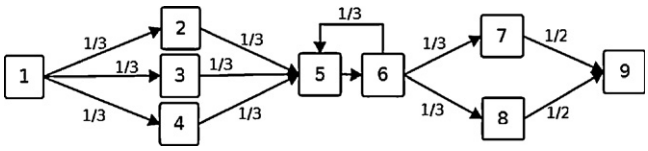



**Fig. 5.** Weighted graph representation of $V_0$ in Fig. 3(a).

The similarity of all edges (using $w_1^0$ and $w_j^1$ as the weights of the edges $e_i^0 \in E_0$ and $e_j^1 \in E_1$) is defined as:

$$\frac{\sum_{i=1}^{\#E_0} \max_{e_j^1 \in E_1}(w_i^0 \cdot w_j^1 \cdot sim_{ed}(e_i^0, e_j^1)) + \sum_{i=1}^{\#E_1} \max_{e_j^0 \in E_0}(w_i^1 \cdot w_j^0 \cdot sim_{ed}(e_i^1, e_j^0))}{\#E_0 + \#E_1} \quad (6)$$

The overall similarity between two process models is defined as a weighted sum of the similarity between activities as defined by Eq. 4 and the similarity between edges as defined by Eq. 6.

*Calculation settings and results:* Table 8 shows the results of the similarity calculations between models $V_0$ and $V_1 \ldots V_7$ when using method 1 for transforming a PM into a graph. Thus, we do not take types of connectors into account and focus solely on equivalence of activities and edges between these activities. Taking the type of connectors into account would result in a lower values for $sim(V_0, V_1)$ and $sim(V_0, V_2)$. The value $sim(V_0, V_3)$ would be higher, since connectors would be included in the calculation.

In the approach of Huang et al., some difficulties occur. While they specify weights for XOR and AND connectors, OR connectors are not taken into account. Therefore, we have used an approach similar to 6.3.1 and weight edges resulting from OR connectors with $\frac{1}{2}$.

*Discussion:* An obvious shortcoming of the approach by Minor et al. using approximation method 1 is that any information about the control flow routing constructs in the model is lost in the transformation: A parallel execution of two activities will be covered in the same way as an exclusive choice among those activities. Methods 2 and 3 have another severe problem: They unite for example all XOR-joins that can be found in a model in the same edge label (method 2) or in the same node (method 3). For example, the model variant $V_4$ (Fig. 3(e)) will be transformed into the same approximation graph as the model $V_0$ (Fig. 3(b)). Consequently, the similarity between both models will be 1 despite of the fact that both models show clearly different behaviours.

As can be seen in the Table 9 the similarity values following the approach presented in [27] are very close to each other. This is due to the fact, that the approach allows an n:m mapping of edges based on the simple comparison of their connected nodes. However, the decisive disadvantage of the approach from [27] is that it does not calculate a similarity of 1 for equivalent models. This drawback results from the multiplication of weights (which is 1 only for sequential activities). Accordingly, this approach ranks model $V_5$ as very similar, since all the weights in this model are 1.

### 6.2. Edit distance between graphs

#### 6.2.1. Graph edit distance similarity

Dijkman et al. [4,28] try to capture structural similarity as follows: As described in Section 6.1.2, they derive a mapping function *map* from a function *corr* that measures the similarities between nodes in $A_0$ and nodes in $A_1$. The nodes $a_0 \in A_0$ for which $map(a_0)$ is not defined and the nodes $a_1 \in A_1$ for which there is no $a_0 \in A_0$ such that $map(a_0) = a_1$ are regarded as "inserted or deleted nodes" (because they appear in one model and not in the other one). Similarly, an edge $(x, y) \in E_0$ is called "inserted or deleted edge" if either $map(x)$ or $map(y)$ is undefined or $(map(x), map(y)) \notin E_1$. Inserted or deleted edges in $E_1$ are defined analogously. With *sn* being the set of inserted or deleted nodes and *se* being the set of inserted or deleted edges, Dijkman et al. define a graph edit distance as:

$$dist(M_0, M_1) = \#sn + \#se + 2 \sum_{a \in A_0, map(a) \text{ is defined}} corr(a, map(a)).$$

By dividing the terms in the above sum by the total numbers of nodes, arcs and nodes that are not inserted or deleted nodes resp., three quotients can be derived. A similarity measure called graph edit distance similarity is calculated as the weighted average of these three quotients.

The idea of a graph-edit distance is also used for comparing processes by Grigori et al. [29]. They use a distance measure for searching a service repository for services that match a given query. The basic ideas for the measure are the same as described above; but two remarkable differences should be noted: First, Grigori et al. do not relate the number of change operations to the graph size and thus violate Property 5. And second, the approach supports a non-injective mapping function *map* which is helpful when models on different levels of abstraction have to be compared.

*Calculation settings and results:* For the similarity calculation according to [4] we used the approach presented in [28]. In doing so, process models are transformed to graph models where connectors are removed and ignored. If connectors are taken into account, $sim(V_0, V_1)$ and $sim(V_0, V_2)$ will be lower (Dijkman et al. state it even may be "too low"). Similarity between individual nodes (the function *corr*) is defined by the Levenshtein string edit distance. As proposed in [4], it is possible to assign individual weights to the summands resulting in different significance for substituted nodes, added and deleted nodes, and added and deleted edges. In our calculation we assign a weight of 1 to every

**Table 8**
Adherence to properties and similarity values of [26].

| Similarity based on common nodes and edges [26] (using approximation method 1) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – no | 3a – no | 5 – yes | 6 – yes | 7 – yes | 8 – yes |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 1.00 | 0.40 | 0.95 | 0.58 | 0.76 | 0.79 |

**Table 9**
Adherence to properties and similarity values of [27].

| Similarity score based on common nodes and weighted edges [27] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – no | 3a – no | 5 – yes | 6 – yes | 7 – yes | 8 – yes |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 0.60 | 0.60 | 0.63 | 0.48 | 0.58 | 0.63 | 0.59 | 0.56 |

**Table 10**
Adherence to properties and similarity values of [4].

| Graph edit distance [4] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – yes | 3a – no | 5 – yes | 6 – yes | 7 – yes | 8 – yes |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 1.00 | 0.63 | 0.97 | 0.73 | 0.86 | 0.88 |

**Table 11**
Adherence to properties and similarity values of [29].

| Graph edit distance [29] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – yes | 3a – no | 5 – no | 6 – no | 7 – yes | 8 – yes |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 0.05 | 0.04 | 0.20 | 0.33 | 0.03 | 0.33 | 0.17 |

component of the calculation resulting in the following equation:

$$sim(M_0, M_1) = 1 - \frac{\frac{\#sn}{\#A_0+\#A_1} + \frac{\#se}{\#E_0+\#E_1} + \frac{2\cdot\sum_{a\in A_0}(1-corr(a,map(a)))}{\#A_0+\#A_1-\#sn}}{3} \quad (7)$$

In Table 11, the similarity values for the approach presented in [29] are shown. To calculate similarity, we make use of the following edit operations, where the last two edit operations result from the possibility to establish non-injective mappings:

- substitute the label of a node
- substitute the label of an edge
- delete a node, connect preceding with succeeding nodes of the deleted node
- delete an edge
- insert an edge
- split a node into two nodes
- combine two nodes into a single one

*Discussion:* Table 10 shows the similarity values established by the calculation given above and the adherence to the properties given in Section 3. Since we transform PMs into their graph representation using method 1 from Fig. 4, $sim(V_0, V_1) = 1$ and $sim(V_0, V_2) = 1$, too.

Since the approach of Grigori et al. considers connectors and their types, the values for $sim(V_0, V_1)$ and $sim(V_0, V_2)$ are rather low in comparison to the other similarity values. Due to the possibility to split a node into two nodes, $sim(V_0, V_3)$ is high because we only have to split node 1 into nodes $<1, A>$, node 5 into nodes $<B, 5>$, node 6 into nodes $<6, C>$, and node 9 into nodes $<D, 9>$.

### 6.2.2. Combining activity matching and a graph edit distance

La Rosa et al. [30] discuss the question of comparing process models stemming from different organisations. Their aim is to create an integrated model in situations like company mergers or restructurings. The approach has three steps: In step 1, a mapping between the activities in $M_0$ and $M_1$ is established, i.e. the mapping function *map* is defined for activity nodes based on a function *corr* that uses string-similarity measures. In step 2, a mapping between split and join nodes is found. For this purpose, a measure called *context similarity* is calculated. A join (or split) node $n_0$ in $M_0$ is regarded as similar to a join/split node $n_1$ in $M_1$, if the mappings (via function *map*) of functions directly preceding and succeeding $n_0$ coincide with the functions directly preceding and succeeding $n_1$. Finally, in step 3, a measure based on a graph-edit distance between $M_0$ and $M_1$ is calculated.

*Calculation settings and results:* To calculate similarity between PMs according to [30], we use a one-to-one-mapping between the sets of nodes (including activities and connectors). However, the type of a connector is not taken into account. The only restriction is that split nodes must not be mapped to join nodes. To calculate similarity we weight every possible edit operation (substitute nodes, add and remove nodes and edges) with an equal weight of 1. The formula to calculate similarity is similar to the one proposed by Dijkman et al. (see Eq. (7) in Section 6.2.1). However, we do not only iterate over activities $A_1$ and $A_2$ but over all nodes $N_1$ and $N_2$ in the PMs.

*Discussion:* The main difference to the approach of Dijkman et al. is that the approach of La Rosa et al. does not transform PMs into their graph representation. Though connectors remain, the type of connectors is not taken into account. For this reason, we have $sim(V_0, V_2) = 1$, but $sim(V_0, V_1) < 1$, i.e. Property 3a does not hold as can be seen in Table 12.

Even if the algorithm is organised in three steps, the information about the matching score between corresponding activities which is calculated in step 1 is not "lost" in later steps; the results from step 1 are included into the final measure. This means that the measure fulfils Property 6.

### 6.2.3. Combining string edit distance and graph edit distance

Another approach that combines the graph-edit distance and the Levenshtein string edit distance [23] for calculating the function *corr* has been published by Kunze and Weske [12]. In short, the graph-edit distance is the least costly sequence of steps to insert or remove a node or to replace a node in $n \in M_0$ by its counterpart $map(n) \in M_1$. The cost of inserting or removing a node or edge is 1, while the cost for replacing activity $n$ by activity $map(n)$ is defined as the Levenshtein string edit distance between $n$ and $map(n)$.

*Calculation settings and results:* To calculate similarity between PMs we transformed the distance measure of [12] into a similarity measure using Eq. (1). Kunze and Weske show the application of their measure on Petri Nets. For our application on generic PMs we distinguish between different connector types. Therefore, for

**Table 12**
Adherence to properties and similarity values of [30].

| Label similarity and graph edit distance [30] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – no | 3a – no | 5 – yes | 6 – yes | 7 – yes | 8 – yes |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 0.81 | 1.00 | 0.83 | 0.96 | 0.62 | 0.92 | 0.94 |

**Table 13**
Adherence to properties and similarity values of [12].

| Combining string edit distance and graph edit distance [12] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – yes | 3a – no | 5 – no | 6 – yes | 7 – yes | 8 – yes |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 0.05 | 0.03 | 0.06 | 0.33 | 0.03 | 0.14 | 0.20 |

**Table 14**
Adherence to properties and similarity values of [31].

| Graph-edit distance by high-level change operations [31] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – no | 3a – yes | 5 – yes | 6 – no | 7 – n/a | 8 – yes |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 0.79 | 0.79 | 0.87 | 0.33 | 0.93 | 0.93 |

transforming $V_0$ into $V_1$, we have to add four new connectors and the respective edges and to transform $V_0$ into $V_2$, we have to remove the XOR connectors and insert OR connectors. The resulting similarity values are shown in Table 13.

*Discussion:* Kunze and Weske show that their distance measure fulfils Properties 1-4, i.e. it is a metric. This allows storing a set of models in a repository organized as a metric tree. For searching a model that is similar to a given query model, it is not necessary to compare the query model with each model from the repository. The main benefit from the paper by Kunze and Weske is their description of the indexing approach based on metric trees which leads to a remarkable improvement of the search for similar models within a model repository.

#### 6.2.4. Graph-edit distance by high-level change operations

Li et al. [31] present an approach to calculate similarity between process models based on so-called high level change operations. They identify different types of high level change operations such as inserting an activity between existing activities, deleting an activity from the model, moving an activity from its original position to another one, and replacing an activity. A high-level change operation encapsulates a number of primitive graph-based operations (deleting an edge, inserting a node etc.). The authors state that by constructing a PM using high level change operations only, it can be guaranteed that the PM is sound. Unfortunately, the paper [31] does not explicitly specify which high-level change operations are supported by the approach.

A distance measure $dist(M_0, M_1)$ is defined in [31] as the minimal number of high-level change operations that is necessary to transform model $M_0 = (N_0, E_0)$ into model $M_1 = (N_1, E_1)$. A corresponding similarity measure is introduced as $sim(M_0, M_1) = 1 - (dist(M_0, M_1)/\#N_0 + \#N_1 - \#(N_0 \cap N_1))$.

*Calculation settings and results:* For our calculations, we referred to the set of high-level change operations described in [19]. Table 14 shows the similarity between $V_0$ and the other PMs as well as the adherence of the measure proposed by Li et al. to the properties given in Section 3. We identified the necessary amount of change operations by hand. For example, to transform $V_0$ into $V_2$ we have to change the types of the four connectors from XOR to OR. Since we also take start and stop events into account, $sim(V_0, V_2) = 1 - (4/(15 + 15 - 11))$. The other similarity values are calculated analogously.

*Discussion:* The similarity measure based on high-level change operations has been developed in the context of the process-aware information system ADEPT [32]. This framework allows to construct sound PMs by starting from an empty model and repeatedly applying high-level change operations. In this context, the question about the difference between model variants $V_0$ and $V_1$ is not relevant, because the construction algorithm would ensure that only one of the models would occur in practice. A remarkable property of this measure is that when calculating $sim(V_0, V_6)$ and $sim(V_0, V_7)$, we have to regard only one high-level change operation. This is different from other similarity measures based on graph edit distances that we have discussed so far.

#### 6.2.5. Tree edit distance between PMs represented as trees

In [33], Bae et al. transform a PM into an ordered tree. A sequential PM (without any splits and joins) would become a tree of depth one; all activities would be leafs that are children of the root node. A split node in the PM would correspond to a node in the tree which is parent of several subtrees which correspond to the outgoing arcs of this split. After translating a PM into a tree this way, algorithms for comparing trees [34] are used.

*Calculation settings and results:* Since the original paper does not describe whether Bae et al. distinguish between different connector types, we transform our models $V_0, \ldots, V_7$ into their graph representation using method 1 from Fig. 4. Therefore, information about connector types gets lost resulting in $sim(V_0, V_1) = 1$ and $sim(V_0, V_2) = 1$, too as can be seen in Table 15. By taking connectors into account, the similarity between $V_0$ and $V_1$ and $V_2$ respectively would be slightly lower.

*Discussion:* The approach described in [33] ignores loops in a PM which is a severe limitation. We cannot agree to the statement made in [33] that "cycles are not used in the distance measure because the cycle does not affect the structure of a process". Additional research would be necessary on extending tree-based similarity measures to the more general case of expressing PMs with loops as trees (preliminaries are discussed in [35,36]).

#### 6.2.6. Edit distance between reduced models

Facilitating queries on process model repositories is in the focus of the approach of Lu and Sadiq in [37]. A query is represented as a partial process model having the desired process structure, e.g. the order of activities. Given a query model $M_0 = (N_0, E_0)$ and a process model $M_1 = (N_1, E_1)$ with the sets $A_0 \subseteq N_0$ and $A_1 \subseteq N_1$ of activities, the mapping $map : A_0 \rightarrow A_1$ is established by label equivalence. The approach is limited by the assumption that $A_0 \subseteq A_1$.

Similar processes can be in either of two relations with each other. $M_1$ is *equivalent* to $M_0$ when $A_0 = A_1$ and $E_0 = E_1$. $M_1$ is *subsumed* by $M_0$ when $A_0 \subseteq A_1$ and the order of activities in $M_0$ is preserved in $M_1$. If models are not in any of those relations, they are not regarded as similar to each other.

To identify the relations it is attempted to transform $M_1$ into the query graph $M_0$ using graph reduction rules. The reduction rules can be found in [38]. Simply stated, the reduction first removes activities from $M_1$ that are not contained in the query graph $M_0$. After these activities are removed, edges that are not required for statements about the order of activities are removed, too. Eventually, $M_1$ will be transformed into a reduced model $M_1^{red} = (N_1^{red}, E_1^{red})$.

**Table 15**
Adherence to properties and similarity values of [33].

| Tree edit distance between PMs represented as trees [33] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – no | 3a – no | 5 – yes | 6 – no | 7 – yes | 8 – yes |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 1.00 | 0.08 | 0.13 | 0.06 | 0.14 | 0.11 |

**Table 16**
Adherence to properties and similarity values of [37].

| Edit distance between reduced models [37] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – no | 3 – no | 3a – no | 5 – yes | 6 – no | 7 – yes | 8 – yes |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 0.82 | 0.82 |

A similarity metric is defined as $sim(M_0, M_1) = \frac{\#E_1^{red} \cap E_0}{\#E_1^{red}}$.

*Calculation settings and results:* To calculate similarity between PMs according to the approach presented by Lu and Sadiq, we make use of the algorithm *SELECTIVE_REDUCE* presented in [38]. This algorithm only distinguishes between split and join connectors regardless of the specific connector types (e.g. AND or XOR).

*Discussion:* The approach of Lu and Sadiq focuses solely on structural similarity between process models based on the order of activities. As can be seen in Table 16, we have $sim(V_0, V_1) = 1$ and $sim(V_0, V_2) = 1$. This is consistent with the algorithm, since it does not distinguish between connector types and unnecessary connectors that do not change the possible order of activities are removed during transformation.

Table 16 also shows that models $V_3$ and $V_4$ get a similarity score of 1 when compared to model $V_0$. This is straightforward, since these variants do not change the order of activities. This behaviour is motivated by the goal of the measure as it is applied during search in process repositories. However, a drawback that cannot be ignored is that the measure of Lu and Sadiq does not take the similarity of activities into account and only counts the amount of common edges.

### 6.3. Causal dependencies between activities

#### 6.3.1. Dependency graph comparison

Bae et al. [39,40] build a so-called "dependency graph" for a PM. The activities of the PM become the nodes in the dependency graph. In the dependency graph, there is an arc between two activities if one activity directly depends on data that have to be produced by another activity, i.e. if one activity is the direct predecessor of another one. For the dependency graph, it does not make a difference which type of connector (AND, XOR, inclusive OR) is located between activities. As an example, the dependency graph of $V_0$ (which coincides with the dependency graphs of $V_1$ and $V_2$) is shown as the topmost graph in Fig. 4 on page 17.

The measure that has been suggested in [39,40] for comparing two dependency graphs $(N_D^0, E_D^0)$ and $(N_D^1, E_D^1)$ is defined as $\#(E_D^0 \setminus E_D^1) \cup (E_D^1 \setminus E_D^0)$, i.e. the number of arcs that exist in one dependency graph, but not in the other one. This way, the (similarity-related) *distance* between the graphs is measured.

The fact that there is no possibility to deal with the semantic meaning of the different types of connectors clearly limits the applicability of this approach. Another shortcoming is illustrated by model variant $V_3$ (Fig. 3(d)): By adding the activities A, B, C and D into model $V_0$, we destroyed the majority of "direct precedence" relations. Therefore, the dependency graphs of $V_0$ and $V_3$ have only one edge (5,6) in common, and the distance measure is equal to the number of all but one edges in both dependency graphs (i.e. 11+15=26). On the other hand, the distance measure between $V_0$ and $V_5$ (whose dependency graphs have two common edges) is 10+6=16 only, which does not meet the intuitive expectation.

In [41], Jung et al. improve the approach with the aim to avoid the shortcomings mentioned above. At first, they calculate the execution probability of each activity. If there is no additional information (for example from process logs), the probability of an activity that follows an XOR-split with $n$ outgoing arcs is assumed to be $\frac{1}{n}$, and the probability of an activity that follows an OR-split is assumed to be $\frac{1}{2}$ regardless of the number of outgoing arcs (which, of course, is disputable). The examples in [41] show only the calculation of execution probabilities for very simple PMs with a nesting level of at most one and without loops. The general case is not described in [41], although a generalization does not seem to be trivial. For comparing two PMs with the activity sets $A_0$ and $A_1$, two vectors are calculated for each model: The activity vector includes the execution probabilities of each activity in $A_0 \cup A_1$. If a model does not include one of these activities, its execution probability is set to 0. The transition vector contains a value for each pair of activities from $(A_0 \cup A_1) \times (A_0 \cup A_1)$. Without loss of generality, we assume that we want to calculate the activity vector for model $M_0$ which has the activity set $A_0$. The value of the transition vector component that corresponds to the pair of activities $(a_I, a_{II})$ is 0 unless activity $a_I$ precedes activity $a_{II}$ in some trace of $M_0$. Otherwise it is the product of the execution probabilities of $a_I$ and $a_{II}$ and the reciprocal of their distance (i.e. the least number of arcs between $a_I$ and $a_{II}$ in the graph $M_0$). For example, the activity vector $a_{v_0}$ of PM $M_0$ in Fig. 3(a) is as follows:

$$a_{v_0} = \left(1 = 1, 2 = \frac{1}{3}, 3 = \frac{1}{3}, 4 = \frac{1}{3}, 5 = 1, 6 = 1, 7 = \frac{1}{2}, 8 = \frac{1}{2}, 9 = 1\right)$$

And respectively, the distance vector $d_{v_0}^1$ and transition vector $t_{v_0}^1$ for activity 1 from $M_0$:

$$d_{v_0}^1 = (2 = 1, 3 = 1, 4 = 1, 5 = 2, 6 = 3, 7 = 4, 8 = 4, 9 = 5)$$

$$t_{v_0}^1 = \left(2 = \frac{1}{3}, 3 = \frac{1}{3}, 4 = \frac{1}{3}, 5 = \frac{1}{2}, 6 = \frac{1}{3}, 7 = \frac{1}{8}, 8 = \frac{1}{8}, 9 = \frac{1}{5}\right)$$

A similarity measure is defined as the cosine of the angle between the activity vector of $M_0$ and the activity vector of $M_1$, and a second similarity measure as the cosine of the angle between both transition vectors.

*Calculation settings and results:* The results of the similarity calculations and the adherence to the properties given in Section 3 for the approach presented in [39] is shown in Table 17. As stated in the description of the measure, dependency graphs do not distinguish between different connector types. Thus, to calculate

**Table 17**
Adherence to properties and similarity values of [39,40].

| Dependency graph comparison [39,40] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – no | 3a – no | 5 – yes | 6 – no | 7 – yes | 8 – yes |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 1.00 | 0.04 | 0.33 | 0.06 | 0.09 | 0.10 |

**Table 18**
Adherence to properties and similarity values of [41].

| Dependency graph comparison [41] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – no | 3a – no | 5 – yes | 6 – no | 7 – yes | 8 – yes |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 0.99 | 0.56 | 0.85 | 0.6 | 0.97 | 0.82 |

similarity, we transform a given PM into its graph representation using approximation method 1 from Fig. 4.

To automatically establish the execution probabilities of activities for the approach of Jung et al., we use the so-called *branch-water* algorithm presented in [42]. As described above, activities following an OR-split are assigned with a probability of $\frac{1}{2}$, activities following an AND-split with a probability of 1, and activities following an XOR-split with a probability of $\frac{1}{n}$ where n is the amount of outgoing arcs. Since Jung et al. do not describe how cycles should be handled, we remove these cycles from our PMs (as this is a necessary requirement for the branch-water algorithm). On the assumption that processes do not run into deadlocks, cycles have no influence on the execution probabilities of individual activities. These preparations result in the similarity values shown in Table 18.

*Discussion:* Since the approach of Bae et al. in [39] does not distinguish between connectors, we have $sim(V_0, V_1) = 1$ and $sim(V_0, V_2) = 1$. Furthermore, when an additional activity is introduced into an existing sequence $A \rightarrow B \rightarrow C \rightarrow D$, a modified model with a sequence $A \rightarrow B \rightarrow C \rightarrow X \rightarrow D$ is regarded as more similar to the original model than a modified model with a sequence $A \rightarrow B \rightarrow X \rightarrow C \rightarrow D$.

Besides challenges in handling cycles, the measure of Jung et al. has additional undesirable properties: Changing an XOR-split with two outgoing arcs into an OR-split does not change the activity and transition vector (though [41] mitigates this effect in presence of execution logs). Furthermore, the consideration of execution probabilities weights a change of an activity within an AND-control block as "more important" than a change of an activity within an XOR-control block.

### 6.3.2. TAR-similarity

In [15], Zha et al. discuss a naive similarity measure based on the set of traces $\Sigma(M_0)$ and $\Sigma(M_1)$.This measure, which they call reference similarity, is defined as $sim(M_0, M_1) = (\#(TAR_0 \cap TAR_1))/(\#(TAR_0 \cup TAR_1))$. Zha et al. name two severe problems of this measure: First, it is not defined for models with loops, because their sets of traces become infinite. And second, it is too rigid as for example the sets of traces of the models shown in Fig. 6 do not share any common trace, i.e. their similarity would be 0.

For these reasons, Zha et al. suggest another measure called TAR-similarity. It is based on the transition adjacency relation (TAR). This relation TAR is defined such that a pair $(A_I, A_{II})$ of two activities belongs to TAR if and only if the model has a trace of the form $\langle \ldots, A_I, A_{II}, \ldots \rangle$, i.e. activity $A_I$ is directly followed by activity $A_{II}$. Let $TAR_0$ be the TAR-relation for model $M_0$ and $TAR_1$ be the TAR-relation for $M_1$. The TAR similarity between $M_0$ and $M_1$ is defined as $sim(M_0, M_1) = (\#(TAR_0 \cap TAR_1)/\#(TAR_0 \cup TAR_1))$ (by using this simple notation, we assume that *map* is the identity). Because the number of activities in $M_0$ and $M_1$ is finite, $TAR_0$ and $TAR_1$ are finite as well, and the measure is defined for all models $M_0$ and $M_1$. In [15], it was shown that the distance measure that is derived from the TAR similarity is a distance function in a metric space.

*Calculation settings and results:* Table 19 shows the resulting similarity values for the approach presented in [15]. To establish TAR sets of PMs having only XOR connectors, it is sufficient to transform these PMs into their graph representation and to use edges between activities. However, if a model contains AND or OR connectors as well, it is necessary to calculate the whole set of possible traces and to extract the TAR sets from these traces.

*Discussion:* The TAR similarity can be seen as an improvement of the measure proposed by Bae et al. [39,40] (see Section 6.3.1), because it distinguishes between different types of splits and joins. However, just as the measure by Bae et al. it has a handicap that can be illustrated by a comparison of the models $V_0$, $V_3$ and $V_5$: The TAR-sets of $V_0$ and $V_3$ have only the element (5,6) in common, while the TAR-sets of $V_0$ and $V_5$ have two elements in common ((6,5) and (1,4)). So we have $sim(V_0, V_5) > sim(V_0, V_3)$, which would not be the expected result. The reason for such results lies in the fact that the TAR relation contains only information about *direct* precedence. It should be appealing to include information about the transitive closure of TAR into the calculation of similarity measures. Instead of analysing information such as "activity $A_I$ can be followed directly by activity $A_{II}$", we would also take into account information such as: "After executing activity $A_I$, it will be possible to execute $A_{II}$ later". Approaches which use this kind of information are discussed in the following subsections.

### 6.3.3. Causal behavioural profiles

Weidlich et al. [43,44] capture the behaviour of a PM by examining dependencies between the execution of an activity $A_I$ and the execution of activity $A_{II}$. Such dependencies are expressed by means of four relations:

- $A_I$ and $A_{II}$ are in strict order relation, if and only if it is possible that $A_I$ is executed before $A_{II}$ is executed, but it is not possible that $A_{II}$ is executed before $A_I$ is executed (i.e., there is a trace $\langle \ldots A_I \ldots A_{II} \ldots \rangle$ but no trace $\langle \ldots A_{II} \ldots A_I \ldots \rangle$
- $A_I$ and $A_{II}$ are in exclusiveness relation, if and only if it is not possible that both $A_I$ and $A_{II}$ are executed in the same process instance.

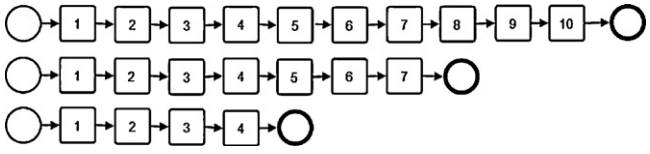



**Fig. 6.** The sets of traces of those models are disjoint.

**Table 19**
Adherence to properties and similarity values of [15].

| Reference similarity [15] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – no | 3a – yes | 5 – yes | 6 – no | 7 – yes | 8 – no |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | | | | not defined ($V_0$ has a loop!) | | | | |
| **TAR-similarity [15]** | | | | | | | | |
| Adherence to property ... | 1 – yes | 2 – yes | 3 – no | 3a – no | 5 – yes | 6 – no | 7 – yes | 8 – no |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 0.60 | 0.04 | 0.85 | 0.11 | 0.41 | 0.47 |


**Fig. 7.** Two models with different sets of traces which have the same causal behavioural profile.

- $A_I$ and $A_{II}$ are in observation concurrency relation (alternatively called "interleaving order relation" in [43]), if and only if it is possible that $A_I$ is executed before $A_{II}$ is executed, and it is also possible that $A_{II}$ is executed before $A_I$ is executed.
- $A_I$ and $A_{II}$ are in co-occurrence relation, if and only if in every process instance for which $A_I$ is executed, $A_{II}$ has to be executed as well.

The set of these four relations is called causal behavioural profile.

For comparing the behaviour modelled by a PM $M_0$ and the behaviour modelled by another PM $M_1$, at first a mapping between "corresponding" activities in $M_0$ and activities in $M_1$ is established. This mapping is expressed by means of a correspondence relation $\sim$. Other than the function *map* discussed in Section 2.3, the correspondence relation $\sim$ has to be neither injective nor functional, i.e. one node in $M_0$ can correspond to multiple nodes in $M_1$ and vice versa.

Weidlich et al. define a similarity metric (called degree of consistency in [43,44]) between $M_0$ and $M_1$ based on the following idea: For each pair of activities in one model for which there are "corresponding" activities in the other model (according to $\sim$), it is checked whether these pairs share the same relations as defined above (strict order, exclusiveness, observation concurrency and co-occurrence).

Formally, let $A_0 \subseteq N_0$ and $A_1 \subseteq N_1$ be the sets of activities in $M_0$ and $M_1$ resp. Then $A_0^{\sim}$ is defined as $\{a \in A_0 : \exists\, b \in A_1$ such that $a \sim b\}$, and $A_1^{\sim}$ is defined as $\{b \in A_1 : \exists\, a \in A_0$ such that $a \sim b\}$. Note that because 1:n mappings are allowed, $\#A_0^{\sim}$ is not necessarily equal to $\#A_1^{\sim}$. Furthermore, the set of consistent pairs $C_0 \subseteq A_0^{\sim} \times A_0^{\sim}$ is defined as all those pairs $(x_0, y_0) \in A_0^{\sim} \times A_0^{\sim}$ for which for all pairs $(x_1, y_1) \in A_1^{\sim} \times A_1^{\sim}$ with $x_0 \sim x_1$ and $y_0 \sim y_1$, $x_0$ and $y_0$ are in the same relations (strict order, exclusiveness, observation concurrency and co-occurrence) as their counterparts $x_1$ and $y_1$. $C_1 \subseteq A_1^{\sim} \times A_1^{\sim}$ is defined analogously.

Using this formalisation, a similarity measure (called consistency metric) can be expressed as $sim(M_0, M_1) = \frac{\#C_0 + \#C_1}{\#(A_0^{\sim} \times A_0^{\sim}) + \#(A_1^{\sim} \times A_1^{\sim})}$.

*Calculation settings and results:* Table 20 shows the similarity values for the approach of Weidlich et al. with taking strict order relations, exclusiveness relations, co-occurrence relations, and interleaving order relations into account. In [44], an efficient algorithm for calculating a process models causal behavioural profile is given. The approach works for sound free-choice Petri nets, i.e. it cannot be used for comparing unsound business process models or models containing OR-splits. Therefore, we had to establish the causal behavioural profile for model variant $V_2$ manually according to the description above. No further adaptations were necessary to calculate the similarity values.

*Discussion:* As shown in [43], the consistency metric can be used for searching pairs of models that can be integrated together into one larger model in order to reduce redundancy in a large model collection. Another important use case (discussed in [43]) is consistency checking between a PM used as a specification and a workflow model used as an implementation. Often, the granularity of the specification differs from the granularity of the implementation. Therefore, for this use case it is a reasonable approach to regard only those activities that have a counterpart in the other model. The others (that do not appear in the pairs of the $\sim$ relation) do not contribute to the calculation of the consistency metric. For this reason, the consistency metric between the models in Fig. 7 would be the same for each pair of models (assuming that $a \sim b$ if and only if $a$ and $b$ have the same label). For other purposes than comparing a specification and an implementation, this could be an undesirable property of this measure.

Note that from the fact two models $M_0$ and $M_1$ have the same causal behavioural profile, it does not follow that $\Sigma(M_0) = \Sigma(M_1)$, as can be seen from the example in Fig. 7.

#### 6.3.4. Causal footprints

Dijkman et al. [4] propose to use causal footprints for capturing the precedence relations between activities in a PM. A causal footprint of a PM with the set of activities $A$ is a pair $(L_{lb}, L_{la})$. The first member of this tuple, $L_{lb} \subseteq (\wp(A) \times A)$ is called the set of look-back links. A pair $(S, a)$ belongs to $L_{lb}$ if and only if each occurrence of activity $a$ in a trace of the PM must be preceded by the occurrence of an activity that is contained in the set $S$. For example, for model $V_0$ in Fig. 3(a), we find:

**Table 20**
Adherence to properties and similarity values of [43].

| Causal behavioural profiles [43] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – no | 3a – no | 5 – yes | 6 – no | 7 – no | 8 – yes |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 0.93 | 0.63 | 0.93 | 0.22 | 0.98 | 0.89 |

- $(\{1\}, 6) \in L_{lb}$ (every occurrence of 6 must be preceded by the occurrence of 1),
- $(\{2, 3, 4\}, 6) \in L_{lb}$ (every occurrence of 6 must be preceded by the occurrence of one activity from the set {2, 3, 4}

According to the definition of $L_{lb}$, we also find that

- $(\{1, 9\}, 6) \in L_{lb}$ (every occurrence of 6 must be preceded by the occurrence of one activity from the set {1, 9}; the presence of "9" is in fact irrelevant, but allowed according to the definition).

Analogously, the set $L_{la}$ of look-ahead links is defined such that $(A \times \wp(A)) \supseteq (a, S) \in L_{la}$ if and only if each occurrence of activity $a$ in a trace of the PM must be *followed* by an occurrence of an activity that is contained in the set $S$.

For measuring the similarity of two PMs, a similarity measure for their causal footprints is calculated. For this purpose, the causal footprints are regarded as documents in a document vector space [45], a concept that is widely used in the field of information filtering and information retrieval. Causal footprints (the "documents") are represented as vectors of index terms. Let's assume that we have to calculate the similarity between the causal footprints of two models $M_0 = (N_0, E_0)$ and $M_1 = (N_1, E_1)$ whose sets of look-ahead links be $L_{la}^{M_0}$ and $L_{la}^{M_1}$ and whose sets of look-back links be $L_{lb}^{M_0}$ and $L_{lb}^{M_1}$.

The set of index terms is defined as $\Theta = N_0 \cup N_1 \cup L_{la}^{M_0} \cup L_{la}^{M_1} \cup L_{lb}^{M_0} \cup L_{lb}^{M_1}$, i.e. $\Theta$ contains all nodes as well as all look-ahead and look-back link of both $M_0$ and $M_1$. Let $\lambda : \Theta \to \mathbb{N}$ be an indexing function that assigns a running number to each index term.

The model $M_i$ $(i \in \{0, 1\})$ is represented as a vector $\vec{g^i} = (g^i_1, g^i_2, \ldots, g^i_{\#\Theta})$. In [46], its coordinates are defined as:

$$g^i_{\lambda(t)} = \begin{cases} 0 & \text{if } t \notin N_i \cup L_{la}^{M_i} \cup L_{lb}^{M_i} \\ 1 & \text{if } t \in N_i \\ \dfrac{1}{2^{len(t)} - 1} & \text{if } t \in L_{la}^{M_i} \cup L_{lb}^{M_i} \end{cases}$$

where $len(t)$ is the number of set elements in the look-ahead or look-back link. For example, $len((\{1\}, 12)) = \#\{1\} = 1$ and $len((\{9, 10, 11\}, 12)) = \#\{9, 10, 11\} = 3$. This way, a greater weight is given to the look-back link $(\{1\}, 12)$, following the rationale that links with fewer activities in the set are more informative and therefore more important for the comparison. The similarity of $M_0$ and $M_1$ is calculated as the cosine of the angle between the corresponding vectors $\vec{g^0}$ and $\vec{g^1}$:

$$sim(M_0, M_1) = \frac{\vec{g^0} \times \vec{g^1}}{\vec{g^0} \cdot \vec{g^1}}.$$

It is easy to see that $sim(M_0, M_1) = 0$ unless $M_0$ and $M_1$ share some common nodes. This means that the above formulae do not explicitly refer to the mapping function *map*; it is simply assumed that *map* is the identity. In [4], the coordinates are calculated in a more sophisticated manner. It is assumed that for generating a mapping function *map*, a similarity measure $corr : N_0 \times N_1 \to [0, 1]$ that compares nodes in $M_0$ with nodes in $M_1$ has been calculated in a preliminary step (cf. Section 6.1.2). The values calculated by this similarity function are preserved when the coordinates of the vectors are calculated. For the purpose of comparing model $M_0$ with model $M_1$, the model $M_0$ is represented as a vector $\vec{g^0} = (g^0_1, g^0_2, \ldots, g^0_{\#\Theta})$ with

$$g^0_{\lambda(t)} = \begin{cases} corr(t, map(t)) & \text{if } t \in N_0 \text{ and } map(t) \text{ is defined} \\ \dfrac{corr(a, map(a))}{2^{len(t)} - 1} & \text{if } L_{la}^{M_0} \ni t = (a, M) \text{ or } L_{lb}^{M_0} \ni t = (M, a) \\ 0 & \text{otherwise} \end{cases}$$

*Calculation settings and results:* Using the reference implementation of Causal Footprints provided as a plug-in to the ProM framework [21], we calculated the similarity values shown in Table 21.

*Discussion:* The advantage of the latter approach is that information about the similarity of nodes (in particular label similarity) that is included in the measure *corr* will be preserved in the similarity measure *sim* that compares the models. This way, for the models shown in Fig. 2, the desirable relation $s(M_0, M_2) > s(M_0, M_1)$ can be achieved (if the function *corr* that compares the activity labels is good enough), i.e. Property 6 holds.

We see the most remarkable disadvantage of the causal footprints in the form described in [46,4] in the fact that $L_{la}$ and $L_{lb}$ contain a very large number of "useless" elements. For example, for model $V_0$ in Fig. 3(a), the look-back links $(\{1\}, 6)$ and $(\{2, 3, 4\}, 6)$ contain substantial information about the relationship between activity 6 and other activities. However, $L_{lb}$ contains additional pairs such as no less than $2^9 - 1$ pairs $(\{1\} \cup S, 6)$ for each subset $S$ of the set of nodes $N$. Constructing the vectors in such a high dimension can become computationally inefficient. A straightforward improvement of the approach would be to consider only look-back/look-ahead links for which no "smaller" look-back/look-ahead link exists. This means that for look-back links, it should be required that $(S, a) \in L_{lb}$ implies that there is no $S' \subset S$ with $(S', a) \in L_{lb}$. An analogous requirement should be set for $L_{la}$.

Another, less severe, disadvantage lies in the fact that in the published algorithm for calculating causal footprints OR-connectors are dealt with in the same way as XOR-connectors. Hence, the change between model variants $V_0$ and $V_2$ will remain undetected; $sim(V_0, V_2)$ is 1. A possible solution of this problem would be to consider other types of look-ahead/look-back links such as $L'_{la}$ as the set of all pairs $(a, S)$ such that every execution of activity $a$ *can* be followed by a state where *all* activities in $S$ are running in parallel.

### 6.3.5. String edit distance of sets of traces

In [47] Wombacher and Rozie compare several approaches to calculate the similarity of process models based on a comparison of their sets of traces. First, they analyse the Levenshtein string edit distance [23] between traces. However, a set of traces of process models with loops is infinite. Thus, this simple idea is not applicable.

To handle infinite traces as well, [47] presents a second approach based on n-grams. These n-grams are defined as subtraces of length n.

For example, possible traces from process variant $V_0$ are ⟨1, 2, 3, 4, 5, 6, 7, 8, 9⟩ and ⟨1, 2, 3, 4, 5, 6, 5, 6, 5, 6, 5, 6, 7, 8, 9⟩. In a trace of $V_0$, activities 5 and 6 can be repeated arbitrarily often. A bigram representation of the traces combines tuples of pairs and is { * 1, 12, 23, 34, 56, 65, 67, 78, 89, 9 * } where * symbolises the start and end of a trace respectively. From the example, we can see that even infinite traces introduced by cycles can be represented using a finite set of n-grams [48].

Analogously to the simple approach, the distance between processes is calculated using the string edit distance. But instead of analysing specific traces only their n-gram-representation is taken into account.

*Calculation settings and results:* Wombacher and Rozie do not give any information on how to calculate the edit distance. Therefore, we calculate the minimum possible edit distances between two bigram representations. Since we only have one-letter activities, the edit distance between bigrams $(a_1, b_1)$ and $(a_2, b_2)$ is either 0 ($map(a_1) = a_2$ and $map(b_1) = b_2$), 1 ($map(a_1) = a_2$ xor $map(b_1) = b_2$), or 2 (neither $a_1$ nor $b_1$ are mapped to $a_2$ or $b_2$).

*Discussion:* Using a bigram representation of process models is similar to the TAR-approach (see Section 6.3.2). As stated there, it

**Table 21**
Adherence to properties and similarity values of [4].

| Causal footprints [4] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – no | 3a – no | 5 – yes | 6 – no | 7 – yes | 8 – no |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 1.00 | 0.45 | 0.80 | 0.59 | 0.97 | 0.84 |

**Table 22**
Adherence to properties and similarity values of [47].

| Sets of traces as n-grams [47] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – no | 3 – no | 3a – no | 5 – no | 6 – no | 7 – yes | 8 – no |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 0.06 | 0.05 | 0.33 | 0.06 | 0.17 | 0.14 |

takes only information about direct precedence of activities into account. Therefore, the similarity values for $(V_0, V_2)$ and $(V_0, V_3)$ are very low in comparison with the other values shown in Table 22.

### 6.4. Approaches based on the sets of traces

#### 6.4.1. Longest common subsequence of traces

To calculate compliance and maturity of an actual process model to a reference model, Gerke et al. [49] compare the sets of traces of both models. In this context, compliance is the extent to which a process model adheres to ordering rules of activities (e.g. activity $A$ must always be executed before activity $B$). Maturity measures to what extent the process model recalls activities of the reference model. In order to avoid problems with infinite traces and infinite sets of traces, it is assumed that the possible executions are restricted by the constraint that there is a maximum number of possible repetitions of each loop in a model.

Gerke et al. use a non-injective mapping function $map : N_0 \rightarrow N_1$. By allowing to map more than one element of $N_0$ to the same element of $N_1$, granularity differences between two models can be handled. Gerke et al. define the compliance degree $cd_{trace}$ and the maturity degree $md_{trace}$ of two traces $\sigma_0$ and $\sigma_1$ based on the length of their longest common subsequence (see Section 2.2) as:

$$\begin{aligned} cd_{trace}(\sigma_1, \sigma_0) &= \frac{len(lcs(\sigma_0, \sigma_1))}{\#\sigma_1} \\ md_{trace}(\sigma_1, \sigma_0) &= \frac{len(lcs(\sigma_0, \sigma_1))}{\#\sigma_0} \end{aligned} \tag{8}$$

The overall compliance and maturity degree between two models $M_0$ and $M_1$ are calculated by summing up the maximum compliance and maturity degree of traces.

$$\begin{aligned} cd(M_1, M_0) &= \frac{\sum_{\sigma_1 \in \Sigma(M_1)} \max_{\sigma_0 \in \Sigma(M_0)} (cd_{trace}(\sigma_1, \sigma_0))}{\#\Sigma(M_1)} \\ md(M_1, M_0) &= \frac{\sum_{\sigma_0 \in \Sigma(M_0)} \max_{\sigma_1 \in \Sigma(M_1)} (cd_{trace}(\sigma_1, \sigma_0))}{\#\Sigma(M_0)} \end{aligned} \tag{9}$$

*Calculation settings and results:* In our example calculations, we use the arithmetic mean between compliance degree $cd(M_1, M_0)$ and maturity degree $md(M_1, M_0)$ as similarity measure. Additionally, we restricted the amount of iterations to 1. Taking more iterations into account would result in slightly higher similarity values between $V_0$ and the other models with iterations ($V_1$, $V_2$, $V_3$, $V_6$, and $V_7$). Accordingly, similarity values between $V_0$ and $V_4$ and $V_5$ would be slightly lower. The results of our calculations can be seen in Table 23.

*Discussion:* By allowing non-injective *map* functions, the approach would lead to a similarity measure *sim* for which for our example model variants $sim(V_0, V_3) = 1$ would hold if *map* is constructed accordingly (we could map 1 to $(1, A)$, 5 to $(B, 5)$, 6 to $(6, C)$, and 9 to $(D, 9)$). [49] describes only the case that one activity in a reference model $M_0$ corresponds to more than one activity in $M_1$. The more general case that $n$ activities in $M_0$ correspond to $m$ activities in $M_1$ is not discussed, i.e. $sim(V_3, V_0)$ would not be defined. It is, however, not difficult to generalize the approach in order to deal with such cases as well.

Above, we have described only a "general" similarity measure that compares two models as a whole. The approach described in [49] supports two more aspects that are important for judging about the compliance of a model with a given reference model: First, it allows that only a subset of the reference model is taken into consideration for the comparison. And second, it is possible to select subsets of activities in the reference model for which the order between them is regarded as unimportant for the comparison. Both aspects are useful for the purpose of measuring the compliance of a PM with a given reference model (which is the aim of Gerke et al. in [49]), although it should be noted that disregarding the order between some activities has a great negative influence on the computational complexity.

In order to compare two models $M_0$ and $M_1$, both sets of traces $\Sigma(M_0)$ and $\Sigma(M_1)$ have to be calculated, and each $\sigma_0 \in \Sigma(M_0)$ has to be compared with each $\sigma_1 \in \Sigma(M_1)$. For models with large sets of traces, this would not be feasible, i.e. we have to observe a violation of Property 8.

#### 6.4.2. Similarity based on principal transition sequences

In order to deal with the problems of infinite traces and infinite sets of traces, Wang et al. [50] limit the (sub)traces to consider in a comparison between PMs as follows: A trace that does not contain any activity more than once is considered as a whole. A trace $\sigma$ that

**Table 23**
Adherence to properties and similarity values of [49].

| Longest common subsequence of traces [49] (with number of loop cycles=1) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – no | 3a – no | 5 – yes | 6 – no | 7 – yes | 8 – no |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 0.86 | 0.79 | 1.00 | 0.43 | 0.93 | 0.90 |

**Table 24**
Adherence to properties and similarity values of [50].

| Similarity based on principal transition sequences [50] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – no | 3a – no | 5 – yes | 6 – no | 7 – yes | 8 – no |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 0.83 | 0.61 | 0.84 | 0.20 | 0.85 | 0.83 |

contains an activity $x$ more than once has the form $\sigma = \langle \sigma_{prefix}, x, \sigma_{repeatable}, x, \dots \rangle$, where $\sigma_{prefix}$ and $\sigma_{repeatable}$ are sub-traces of $\sigma$. In such a case, the sub-traces $\sigma_{prefix}$ and $\sigma_{repeatable}$ are used for the comparison instead of the complete trace $\sigma$. In [50] it has been shown that for each PM the number of (sub)traces derived in this way is finite.

The similarity between two (sub)traces $\sigma_I$ and $\sigma_{II}$ is defined based on the longest common subsequence as

$sim_{trace}(\sigma_I, \sigma_{II}) = \frac{len(lcs(\sigma_I, \sigma_{II}))}{max(len(\sigma_I), len(\sigma_{II}))}$. Based on this formula, the similarity between two PMs $M_0$ and $M_1$ with the sets of (sub)traces $T_0$ and $T_1$ selected according to the above description is defined as

$$sim(M_0, M_1) = \frac{\sum_{\sigma \in T_0} \max_{\sigma' \in T_1} sim_{trace}(\sigma, \sigma') + \sum_{\sigma' \in T_1} \max_{\sigma \in T_0} sim_{trace}(\sigma', \sigma)}{\#T_0 + \#T_1} \quad (10)$$

*Calculation settings and results:* The values for the similarity between $V_0$ and the rest of our example variants according to [50] is shown in Table 24.

*Discussion:* The approach from Wang et al. allows to calculate a similarity measure based on sets of traces even between PMs with an infinite sets of traces. However, the generation of the sub-traces to compare still requires a symbolic exploration of the sets of traces, i.e. Property 8 is not fulfilled.

### 6.4.3. Similarity of process models based on observed behaviour

De Medeiros et al. [51] present a method to calculate the similarity of PMs that is based on comparing traces obtained from actual process executions or by simulation. They point out that comparing the sets of traces directly leads to problems when a set of traces becomes infinite and that such a comparison would not take into account the real world application of processes in practice where certain traces occur more frequently than others. It could be added that dealing with the whole sets of traces could become computationally inefficient if the sets of traces are very large.

De Medeiros et al. define a log $L$ as a set of traces of a PM together with their occurrence frequency $L(\sigma)$. The frequency of a trace is used to put more importance to those traces that were observed more frequently.

A partial trace $p(\sigma, k)$ of a trace $\sigma$ represents the first $k$ activities in this trace. The set of enabled activities of a model $M_0$ (denoted as $e(M_0, p(\sigma, k))$) contains all activities that can be executed after the execution of a partial trace $p(\sigma, k)$, i.e. an activity $x$ belongs to $e(M_0, p(\sigma, k))$ if and only if there is a trace that has the form $\langle p(\sigma, k), x, \dots \rangle$

De Medeiros et al. define two similarity measures: The *precision* measure shows the extent to which dependencies between activities in the second model can be found in the first model as well. To simplify calculation we first define the *precision factor p* for two models $M_0$ and $M_1$ based on a trace $\sigma$.

$$p(M_0, M_1, \sigma) = \sum_{i=0}^{\#\sigma - 1} \frac{\#(e(M_0, p(\sigma, i)) \cap e(M_1, p(\sigma, i)))}{\#e(M_1, p(\sigma, i))} \quad (11)$$

Using this factor it is now possible to define the similarity measure "precision" between the models $M_0$ and $M_1$ as

$$sim_{precision}(M_0, M_1) = \frac{\sum_{\sigma \in L} \frac{L(\sigma)}{\#\sigma} \cdot p(M_0, M_1, \sigma)}{\#L} \quad (12)$$

A second measure, the *recall*, shows the extent to which dependencies between activities of the first model can be found in the second model as well. It is defined in the same way as the precision measure except the recall factor is divided by the number of enabled activities in model $M_0$ instead of the number of enabled activities in $M_1$.

*Calculation settings and results:* We calculate the similarity of two process models as the arithmetic mean between recall and precision. Since logs were not available in our study, we use the original process models without executing the iteration. For model $V_0$ we have a set $L$ of six logs:

$$L = \{125679, 125689, 135679, 135689, 145679, 145689\}$$

This leads to the similarity values shown in Table 25. If we would have executed the iteration once or more than once this would have resulted in slightly lower similarity values between models $V_0$ and $V_4$ and $V_5$ and in slightly higher similarity values for the other models.

*Discussion:* The approach of de Medeiros et al. uses logs of executed PMs as input. Therefore, it is only applicable if information about process execution exists. If no logs are available, it would be possible to compute logs by randomly simulating the executions of a PM. Furthermore, the authors themselves state that the approach is limited to comparing two models *with respect to a given log*. For example, if subparts of a model are never executed, they are not in the log and therefore, differences in these subparts can not be identified. For this reason, the authors emphasise that the logs must reflect *typical* behaviours of the models.

### 6.5. Similarity of structural complexity metrics

In this subsection, we shortly mention another approach for defining similarity between PMs that will not be discussed in detail, because it follows a different understanding of the concept of similarity.

Melcher and Seese [52] aim to find structurally similar PMs within model collections by comparing the values of several complexity metrics for the PMs. The models are clustered such that

**Table 25**
Adherence to properties and similarity values of [51].

| Similarity based on traces [51] (taking $\Sigma(M_0)$ as log) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Adherence to property ... | 1 – yes | 2 – yes | 3 – no | 3a – no | 5 – yes | 6 – no | 7 – yes | 8 – yes |
| Similarity between $V_0$ and ... | $V_0$ | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| | 1.00 | 1.00 | 0.90 | 0.33 | 0.83 | 0.22 | 0.72 | 0.65 |

**Table 26**
Similarity measures for our example models.

| | Similarity between $V_0$ and ... | | | | | | |
|---|---|---|---|---|---|---|---|
| | $V_1$ | $V_2$ | $V_3$ | $V_4$ | $V_5$ | $V_6$ | $V_7$ |
| Measures based on the correspondence of nodes and edges (not taking into account the control flow) | | | | | | | |
| Percentage of common activity names [22] | 1.00 | 1.00 | 0.82 | 1.00 | 1.00 | 1.00 | 1.00 |
| Label matching similarity [4] | 1.00 | 1.00 | 0.82 | 1.00 | 1.00 | 1.00 | 1.00 |
| Similarity of activity labels [6] | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Feature-based activity similarity [25] | 1.00 | 1.00 | 0.82 | 1.00 | 1.00 | 1.00 | 1.00 |
| Percentage of common nodes and edges [26] | 1.00 | 1.00 | 0.40 | 0.95 | 0.58 | 0.76 | 0.79 |
| Node- and link-based similarity [27] | 0.60 | 0.63 | 0.48 | 0.58 | 0.63 | 0.59 | 0.56 |
| Measures based on graph edit distances | | | | | | | |
| Graph edit distance [4] | 1.00 | 1.00 | 0.63 | 0.97 | 0.73 | 0.86 | 0.88 |
| Graph edit distance [29] | 0.05 | 0.04 | 0.20 | 0.33 | 0.03 | 0.33 | 0.17 |
| Label similarity and graph edit distance [30] | 0.81 | 1.00 | 0.83 | 0.96 | 0.62 | 0.92 | 0.94 |
| Label similarity and graph edit distance [12] | 0.05 | 0.03 | 0.06 | 0.33 | 0.03 | 0.14 | 0.20 |
| Number of high-level change operations [31] | 1.00 | 0.79 | 0.79 | 0.87 | 0.33 | 0.93 | 0.93 |
| Comparing PMs represented as trees [33] | 1.00 | 1.00 | 0.07 | 0.13 | 0.06 | 0.14 | 0.14 |
| Edit distance between reduced models [33] | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 0.82 | 0.82 |
| Measures that analyse causal dependencies between activities | | | | | | | |
| Comparing dependency graphs [39,40] | 1.00 | 1.00 | 0.04 | 0.33 | 0.06 | 0.09 | 0.10 |
| Comparing dependency graphs [41] | 1.00 | 0.99 | 0.56 | 0.85 | 0.60 | 0.97 | 0.82 |
| Reference similarity [15] | not defined ($V_0$ has a loop!) | | | | | | |
| TAR-relationship [15] | 1.00 | 0.60 | 0.04 | 0.85 | 0.11 | 0.41 | 0.47 |
| Causal behavioural profiles [43] | 1.00 | 0.93 | 0.63 | 0.93 | 0.22 | 0.98 | 0.89 |
| Causal footprints [4] | 1.00 | 1.00 | 0.45 | 0.80 | 0.59 | 0.97 | 0.84 |
| Sets of traces as n-grams [47] | 1.00 | 0.06 | 0.05 | 0.33 | 0.06 | 0.17 | 0.14 |
| Measures that compare sets of traces or logs | | | | | | | |
| Longest common subsequence of traces [49] | 1.00 | 0.86 | 0.79 | 1.00 | 0.43 | 0.93 | 0.90 |
| Similarity based on principal transition sequences [50] | 1.00 | 0.83 | 0.61 | 0.84 | 0.20 | 0.85 | 0.83 |
| Similarity based on traces [51] | 1.00 | 0.90 | 0.33 | 0.83 | 0.22 | 0.72 | 0.65 |

PMs with similar metrics values can be identified. While this could be useful for gaining insights into the distribution of metric values, it is not possible to draw conclusions about behavioural similarity or relatedness among PMs. For this reason, the approach will not be discussed here further.

## 7. Discussion

Table 26 shows the similarity values we have computed between our example model $V_0$ and its variants $V_1 \dots V_7$. For measures that can be parametrised by attaching different weights to factors, we used the most reasonable and simple parameters (e.g. for a measure which is a weighted sum of three numbers, every summand has a weight of $\frac{1}{3}$). Some results do not comply with the intuitive understanding of "PM similarity", for example some measures have $sim(V_0, V_3) < sim(V_0, V_5)$.

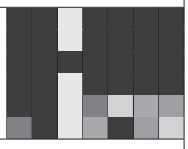
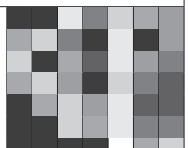
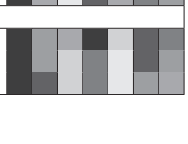
Table 27 allows a more visual comparison between the similarity measures. For each measure, the darkness of a cell represents the order of similarity that the similarity measure computes between $V_0$ and the model variant. We observe that some measures based on graph-distance do not agree with the majority of measures about the value of $sim(V_0, V_1)$. Also, we can see that the measures come to very different results about the similarity between $V_0$ and the models $V_4$, $V_6$ and $V_7$. A rather surprising observation is that there are measures that regard $V_5$ as more similar to $V_0$ than at least one other model.

As stated in Section 4.2, similarity measures for PMs have been proposed for a number of purposes. Our observations give some first insights which measures are more useful than others for a given purpose:

The measures discussed in Section 6.1 which do not take into account the routing constructs in a PM are useful for finding related models from a repository (and less useful for purposes that make reference to the model behaviour). An interesting use case for such rather simple measures has been suggested in [25]: A search for related models can be used as a first step of a search in a large repository. It helps to filter out unrelated models such that the more precise (but also slower) algorithms can be applied to a small subset of the original search space.

Another similarity measure not taking into account the routing constructs is the measure suggested by Bae et al. [39]. The approach is based on a "dependency graph" that documents the precedence and causality relations between activities. Such an approach is suitable for modelling languages such as IDEF0 [53] which show this kind of relations but less useful for languages such

**Table 27**
$sim(V_0, V_i)$ is represented by the darkness of the table cells.

| | Similarity between $V_0$ and ... $\|V_1\|V_2\|V_3\|V_4\|V_5\|V_6\|V_7$ |
|---|---|
| Measures based on the correspondence of nodes and edges (not taking into account the control flow) | |
| Percentage of Common Activity Names [22] |  |
| Label Matching Similarity [4] | |
| Similarity of Activity Labels [6] | |
| Feature-Based Activity Similarity [25] | |
| Percentage of Common Nodes and Edges [26] | |
| Node- and Link-Based Similarity [27] | |
| Measures based on graph edit distances | |
| Graph Edit Distance [4] |  |
| Graph Edit Distance [29] | |
| Label Similarity and Graph Edit Distance [30] | |
| Label Similarity and Graph Edit Distance [12] | |
| Number of High-Level Change Operations [31] | |
| Comparing PMs Represented as Trees [33] | |
| Edit Distance Between Reduced Models [33] | |
| Measures that analyse causal dependencies between activities | |
| Comparing Dependency Graphs [39, 40] |  |
| Comparing Dependency Graphs [41] | |
| TAR-Relationship [15] | |
| Causal Behavioural Profiles [43] | |
| Causal Footprints [4] | |
| Sets of Traces as n-grams [47] | |
| Measures that compare sets of traces or logs | |
| Longest Common Subsequence of Traces [49] |  |
| Similarity Based on Principal Trans. Sequences [50] | |
| Similarity Based on Traces [51] | |

as BPMN that put focus on advanced control flow constructs as well.

When models are compared with the aim of discovering services or measuring conformance, approaches that consider the actual behaviour of a process execution have to be used. Preference should be given to the methods described in Section 6.3 that exploit relationships between activities instead of requiring a calculation of the whole sets of traces as some approaches discussed in Section 6.4 do. The reason is that calculating the whole set of traces of a model can demand large memory and processing resources. It has to be noted that the approach based on causal footprints (Section 6.3.4) in its current form is computationally inefficient as well and cannot be recommended to be used in the context of large PM repositories. Though we use the reference implementation of the causal footprints, it takes about 5 seconds to compute similarity between the eight process variants using our ProM plugin. All other measures calculate these similarity values in less than 1 second.

Processing speed can be less important if only two models have to be compared, for example to measure conformance. In such cases, using approaches that require to calculate the sets of traces can be an option.

Some applications require to compare PMs that have been designed on different levels of granularity. For example, this can be the case if the conformance between a PM serving as a specification and the actual implementation in a workflow system have to be compared. In such cases, it is recommended to use a measure that finds a similarity even between such models. In particular, the approaches of Lu and Sadiq [38] (see Section 6.2.6), Weidlich et al. [43] (see Section 6.3.3), Gerke et al. [49] (see Section 6.4.1) and Grigori et al. [29] (see Section 6.2.1) support such use cases.

Although not extensively discussed in our article, it should be noted that the quality of the mapping between the nodes (the function *map*) has a significant contribution to the quality of a similarity measure. In particular, regarding nodes as corresponding to each other only if they have exactly the same label is reasonable only in a few special application areas such as comparing models that have been derived from the same template.

## 8. Conclusion

The aim of this survey was to discuss the different concepts for defining similarity measures for PMs.

We elaborated a number of desirable properties for PM similarity measures and analysed 23 similarity measures that have been described in the literature with respect to those properties. Also, we computed the similarity between example models using the different similarity measures. The results show that hardly a measure fulfils all desirable properties. Furthermore, it can be seen that different similarity measures rank the similarity between PMs very differently. We conclude that there is not a single "perfect" similarity measure. Instead, we gave some recommendations for the selection of an appropriate similarity measure for different use cases.

We are not aware of any other work that aims to give a comprehensive overview about existing PM similarity measures. We hope that our article is a contribution that helps to improve similarity measures and to promote their practical application.

## References

[1] Gulla, J.A. Brasethvik,T. On the challenges of business modeling in large-scale reengineering projects, in: IEEE International Conference on Requirements Engineering, 2000, p. 17.
[2] Dumas, M. García-Bañuelos, L. Dijkman,R. Similarity search of business process models, IEEE Data Engineering Bulletin 32 (2009) 23–28.
[3] Z. Yan, R. Dijkman, P. Grefen, Business process model repositories—framework and survey, Working Papers 292, Technische Universiteit Eindhoven, Eindhoven, 2009.
[4] Dijkman, R. Dumas, M. van Dongen, B. Käärik, R. Mendling,J. Similarity of business process models: metrics and evaluation, Information Systems 36 (2011) 498–516.
[5] Weidlich, M. Dijkman, R.M. Mendling,J. The ICoP framework: identification of correspondences between process models, in: Proceedings of the 22nd International Conference of Advanced Information Systems Engineering, CAiSE 2010, Hammamet, Tunisia, vol. 6051 of LNCS, Springer, (2010), pp. 483–498.
[6] Ehrig, M. Koschmider, A. Oberweis,A. Measuring similarity between semantic business process models, in: Proceedings of the Fourth Asia-Pacific Conference on Conceptual modelling, vol. 67, 2007, pp. 71–80.
[7] Niemann, M. Siebenhaar, M. Eckert, J. Steinmetz,R. Process model analysis using related cluster pairs, in: Proceedings of the 1st International Workshop Process in the Large (IW-PL'10), Hoboken, NJ, September 2010, 2010.
[8] Object Management Group Business Process Model and Notation, Version 2.0, 2011.
[9] Bergroth, L. Hakonen, H. Raita,T. A survey of longest common subsequence algorithms, in: Proceedings of the International Symposium on String Processing and Information Retrieval, 2000, p. 39.
[10] Santini, S. Jain,R. Similarity measures, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (1999) 871–883.
[11] Lin,D. An information-theoretic definition of similarity, in: Proceedings of the 15th International Conference on Machine Learning, 1998, pp. 296–304.
[12] Kunze, M. Weske,M. Metric trees for efficient similarity search in process model repositories., in: Proceedings of the 1st International Workshop Process in the Large (IW-PL'10), Hoboken, NJ, September 2010, 2010.
[13] Tversky,A. Features of similarity, Psychological Review 84 (1977) 327–352.
[14] Brocke, J.V. Simons, A. Niehaves, B. Riemer, K. Plattfaut, R. Cleven,A. Reconstructing the giant: on the importance of rigour in documenting the literature search process., in: Proceedings of the 17th European Conference On Information Systems, Verona, 2009, pp. 2206–2217.
[15] Zha, H. Wang, J. Wen, L. Wang, C. Sun,J. A workflow net similarity measure based on transition adjacency relations, Computers in Industry 61 (2010) 463–471.
[16] Reiss,S.P. Semantics-based code search, in: ICSE '09: Proceedings of the 2009 IEEE 31st International Conference on Software Engineering, IEEE Computer Society, Washington, DC, USA, (2009), pp. 243–253.
[17] P. Fettke, P. Loos, J. Zwicker, Business process reference models: survey and classification, In: C. Bussler, A. Haller (Eds.), Business Process Management Workshops, Vol. 3812 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 2006, pp. 469–483.
[18] Dijkman,R. A classification of differences between similar business processes, in: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference, IEEE Computer Society, Washington, DC, USA, (2007), pp. 37–50.
[19] Weber, B. Reichert, M. Rinderle-Ma,S. Change patterns and change support features—enhancing flexibility in process-aware information systems, Data and Knowledge Engineering 66 (2008) 438–466.
[20] M. Weidlich, A.P. Barros, J. Mendling, M. Weske, Vertical alignment of process models—how can we get there? In: Proceedings of the 10th International Workshop on Enterprise, Business-Process and Information Systems Modeling, Vol. 29 of LNBIP, Springer, 2009, pp. 71–84.
[21] van Dongen, B. de Medeiros, A. Verbeek, H. Weijters, A. van der Aalst,W. The prom framework: a new era in process mining tool support, in: CiardoG. , DarondeauP. (Eds.), Applications and Theory of Petri Nets 2005. Vol. 3536 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 2005, pp. 1105–1116.
[22] Akkiraju, R. Ivan,A. Discovering business process similarities: an empirical study with SAP best practice business processes, in: Proceedings of the 8th International Conference on Service-Oriented Computing. Vol. 6470 of LNCS, 2010, pp. 515–526.
[23] Levenshtein,V.I. Binary codes capable of correcting deletions, insertions and reversals, Soviet Physics Doklady 10 (1966) 707.
[24] Miller,G.A. Wordnet: a lexical database for English, Communications of the ACM 38 (1995) 39–41.
[25] Yan, Z. Dijkman, R.M. Grefen,P. Fast business process similarity search with feature-based similarity estimation, in: On the Move to Meaningful Internet Systems - Confederated International Conferences Proceedings 2010, Part I. Vol. 6426 of LNCS., Springer, 2010, pp. 60–77.
[26] Minor, M. Tartakovski, A. Bergmann,R. Representation and structure-based similarity assessment for agile workflows, in: Proceedings of the 7th International Conference on Case-Based Reasoning, Springer, 2007, pp. 224–238.
[27] K. Huang, Z. Zhou, Y. Han, G. Li, J. Wang, An algorithm for calculating process similarity to cluster open-source process designs, In: Grid and Cooperative Computing 2004 Workshops, Springer, 2004, pp. 107–114.
[28] Dijkman, R. Dumas, M. García-Bañuelos,L. Graph matching algorithms for business process model similarity search, in: Proceedings of the Business Process Management. Vol. 5701 of LNCS, Springer, 2009, pp. 48–63.
[29] Grigori, D. Corrales, J.C. Bouzeghoub, M. Gater,A. Ranking BPEL processes for service discovery, IEEE Transactions on Services Computing 3 (2010) 178–192.
[30] Rosa, M.L. Dumas, M. Uba, R. Dijkman,R.M. Merging business process models, in: Proceedings of the Confederated International Conferences On the Move to Meaningful Internet Systems, 2010, Part I. Vol. 6426 of LNCS, Springer, 2010, pp. 96–113.
[31] Li, C. Reichert, M. Wombacher,A. On measuring process model similarity based on high-level change operations, in: Proceedings of the 27th International Conference on Conceptual Modeling, Springer, 2008, pp. 248–264.

[32] Reichert, M. Dadam,P. ADEPT flex-supporting dynamic changes of workflows without losing control, Journal of Intelligent Information Systems 10 (1998) 93–129.
[33] J. Bae, J. Caverlee, L. Liu, H. Yan, Process mining by measuring process block similarity, In: Proceedings of the Business Process Management Workshops 2006, Vol. 4103 of LNCS, Springer, 2006, pp. 141–152.
[34] Bille,P. A survey on tree edit distance and related problems, Theoretical Computer Science 337 (2005) 217–239.
[35] Vanhatalo, J. Völzer, H. Koehler,J. The refined process structure tree, in: Proceedings of the 6th International Conference on Business Process Management (BPM), 2008, Milan, Italy, September 2–4, 2008. Vol. 5240 of Lecture Notes in Computer Science, Springer, 2008, pp. 100–115.
[36] Gerth, C. Luckey, M. Küster, J.M. Engels,G. Detection of semantically equivalent fragments for business process model change management, in: Proceedings of the 2010 IEEE International Conference on Services Computing. SCC'10, IEEE Computer Society, Washington, DC, USA, (2010), pp. 57–64.
[37] R. Lu, S.W. Sadiq, On the discovery of preferred work practice through business process variants, ER, 2007, pp. 165–180.
[38] R. Lu, S. Sadiq, On managing process variants as an information resource, Technical report, The University of Queensland, School of Information Technology and Electrical Engineering, 2006.
[39] Bae, J. Liu, L. Caverlee, J. Rouse,W.B. Process mining, discovery, and integration using distance measures, in: Proceedings of the IEEE International Conference on Web Services, 2006, pp. 479–488.
[40] Bae, J. Liu, L. Caverlee, J. Zhang, L.J. Bae,H. Development of distance measures for process mining, discovery and integration, International Journal of Web Services Research 4 (2007) 1–17.
[41] Jung, J.Y. Bae, J. Liu,L. Hierarchical clustering of business process models, International Journal of Innovative Computing, Information and Control 5 (2009) 1–11.
[42] Bae, J. Bae, H. Kang, S.H. Kim,Y. Automatic control of workflow processes using ECA rules, IEEE Transactions on Knowledge and Data Engineering 16 (2004) 1010–1023.
[43] Weidlich, M. Mendling, J. Weske,M. Efficient consistency measurement based on behavioural profiles of process models, IEEE Transactions on Software Engineering 99 (2010).
[44] Weidlich, M. Polyvyanyy, A. Mendling, J. Weske,M. Efficient computation of causal behavioural profiles using structural decomposition, in: Proceedings of the 31st International Conference on Applications and Theory of Petri Nets (PETRI NETS 2010), Braga, Portugal, June 21–25, 2010. Vol. 6128 of Lecture Notes in Computer Science, Springer, 2010, pp. 63–83.
[45] Salton, G. Wong, A. Yang,C.S. A vector space model for automatic indexing, Communications of the ACM 18 (1975) 613–620.
[46] van Dongen, B. Dijkman, R. Mendling,J. Measuring similarity between business process models, in: Proceedings of the Advanced Information Systems Engineering. Vol. 5074 of LNCS, Springer, 2008, pp. 450–464.
[47] Wombacher, A. Rozie,M. Evaluation of workflow similarity measures in service discovery, in: Proceedings of the Service Oriented Electronic Commerce: Proceedings zur Konferenz im Rahmen der Multikonferenz Wirtschaftsinformatik. Vol. 80 of LNI, GI, 2006, pp. 51–71.
[48] Mahleko, B. Wombacher, A. Fankhauser,P. Process-annotated service discovery facilitated by an n-gram-based index, in: proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service on e-Technology, e-Commerce and e-Service. EEE'05, IEEE Computer Society, Washington, DC, USA, (2005), pp. 2–8.
[49] Gerke, K. Cardoso, J. Claus,A. Measuring the compliance of processes with reference models, in: Proceedings of the Confederated International Conferences 2009 On the Move to Meaningful Internet Systems, Part I, Springer, 2009, pp. 76–93.
[50] Wang, J. He, T. Wen, L. Wu, N. ter Hofstede, A.H.M. Su,J. A behavioral similarity measure between labeled petri nets based on principal transition sequences - (short paper), in: Proceedings of the Confederated International Conferences On the Move to Meaningful Internet Systems -2010. Part I. Vol. 6426 of LNCS, Springer, 2010, pp. 394–401.
[51] Alves de Medeiros, A.K. van der Aalst, W.M.P. Weijters,A.J.M.M. Quantifying process equivalence based on observed behavior, Data and Knowledge Engineering 64 (2008) 55–74.
[52] Melcher, J. Seese,D. Visualization and clustering of business process collections based on process metric values, in: International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 2008, 572–575.
[53] Draft Federal Information Processing Standards Publication 183, Integration Definition for Function Modeling (IDEF0), 1993.

**Michael Becker** (born in 1982) studied computer science at the University of Leipzig, Germany. After graduating, he worked at the chair of Telematics and E-Business. Since 2010, he is associated with the department of Business Information Systems at the University of Leipzig. His research interests include connecting software and service engineering and business process management. In addition to his scientific activities, he works as a Typo3 consultant.

**Ralf Laue** (born in 1968) studied mathematics at the University of Leipzig, Germany. After graduating, he worked as a system programmer before returning to the University of Leipzig in 2003. He obtained a PhD in computer science in 2010. Since 2011, he is a full professor for software engineering at the University of Applied Sciences in Zwickau, Germany. His research interests include the application of formal methods on business process models and understandability of visual models.