On the Suitability of Process Model Similarity Metrics for Evaluating Replaceability

Jörg Lenhard

Department of Mathematics and Computer Science,
Karlstad University, 65188 Karlstad, Sweden
joerg.lenhard@kau.se

Abstract. In the field of process-aware information systems, much work has been devoted to developing metrics for determining the similarity between process models. Similarity assessment is important for a wide array of applications and one area that has received relatively little attention so far is the replaceability assessment of executable process models. Replaceability assessment is relevant during software migration, e.g., when upgrading to a new execution platform. In this setting, it might be necessary to replace process models that can no longer be executed on the newer platform. Many of the existing metrics are ill-suited for replaceability assessment, because they were developed for non-executable models and tend to abstract from details that are decisive for the aforementioned application scenario. This paper discusses existing metrics for similarity assessment in a literature review and selects a subset of the body of metrics as candidates for replaceability assessment. Using an exemplary computation, it recommends a particular metric, TAR-similarity, for this purpose. Through this evaluation, this paper can be seen as a motivation for developing better node mapping functions.

Keywords: replaceability, similarity, software metrics, process model

1 Introduction

In the software industry, changes take place at an unprecedented pace. Market pressure forces enterprises to constantly revise and update their IT-systems to maintain competitive advantages and to react to customer demands [4]. The necessity to integrate new technologies or features, whilst keeping existing functionality and coping with increasing load, leads to a need for continuous evolution of system structure with respect to software and hardware [24].

In the area of distributed systems, the trends of service-orientation [27] and process-awareness [8] have emerged to cope with such challenges. Computing systems are built as loosly coupled sets of services, with a trend towards microservices [26], which interact by exchanging messages to provide higher-level functionality. This approach eases system evolution, since, in a well-designed system, singular services can be changed or upgraded without impact on the overall system. Message exchanges between services are oftentimes structured

by capturing them in explicit process representations or *process models* [8]. During the implementation phase of the system [8, p. 12], executable models can be deployed and instantiated on a particular type of middleware, called *process engines*. In this setting, an executable process model corresponds to a piece of application software, whereas the process engine represents the execution environment. Process engines have considerable influence on the runtime quality of service provided by the hosted applications. Consequently, there is value in selecting the best performing engine and migrating to newer engines, as evidenced by approaches on engine benchmarking [11, 12], engine selection [13], or the advent of cloud-based engines with improved performance characteristics [15].

Process engines and models interface via language standards, such as the *Business Process Model and Notation* (BPMN) [17]. A major motivation for standardization is the portability of process models among engines enabled by standards. If a model and a set of engines conform to a standard, the process model can be freely migrated between the engines. In theory, this defends from vendor lock-in and potentially eases system evolution. However, work on standard conformance of engines [12, 14] demonstrates that standard support in engines is very diverse in practice. Despite the existence of standards, engines tend to implement differing subsets of the specified features. Thus, it might not necessarily be feasible to migrate an executable process model to a newer engine, even if it is implemented in a standardized notation. In this case, a straight-forward mode of action, described by software quality standards [16], is the replacement of an application by an alternative one. In other words, if a process model cannot be migrated, it could be considered to replace the existing model with a different model that runs on the newer engine. This replacement scenario is the motivation for this paper.

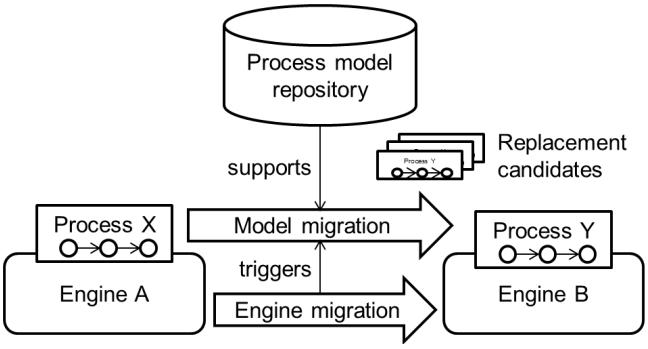

Fig. 1. Migration Scenario Leading to Process Model Replacement

Clearly, a replacement is only possible if alternatives to the original model do exist and they are sufficiently *replaceable* with each other. An eventual source of replacement candidates are process model repositories [35]. Management of such repositories is a common task for enterprises that adopt process-aware and

service-oriented technologies. If replacement candidates are available in such a repository, it is possible to compute the *replaceability* [16] for all paired combinations of the enacted model and its alternatives to determine the best fitting replacement candidate. This migration scenario is visualized in Fig. 1. The migration from *Engine A* to *Engine B* triggers the necessity to migrate *Process X*, which unfortunately cannot be modified to enable execution on *Engine B*. Therefore, it needs to be replaced and this is supported by a *process model repository* that supplies a number of *replacement candidates*, which can be modified to run on *Engine B*. From these candidates, *Process Y* is found to be most replaceable. Therefore *Process X* is replaced by *Process Y* during the migration.

The property of central concern here is denoted as *replaceability* by software quality models [16]. In essence, it translates to *similarity* [38]. If two process models are very similar to each other, they can replace one another with relative ease. This implies that metrics for evaluating process model similarity seem suited for evaluating replaceability and opens up a large space of related work. Many similarity metrics have been proposed in the literature, e.g., [1, 2, 6, 18, 19, 23, 25, 31, 33, 34, 37]. What is more, comparative studies that evaluate these metrics do exist [3]. Thus, the addition of yet another set of replaceability metrics on top of the existing body of metrics is neither desirable, nor likely to form a novel contribution. Instead, in this paper, we provide a discussion of existing similarity metrics for the purpose at hand. We select a subset of metrics based on a categorization from [3] and discuss these metrics more closely. Furthermore, we outline crucial problems that remain for using these metrics to evaluate replaceability, essentially a proper node mapping function, and propose to take node types into account during evaluation. With the help of an exemplary computation, we are, thus, able to identify one metric, TAR-similarity [37], as most suitable for our use case.

The approach applied in this paper has been sketched in [20] and the paper is based on an excerpt of [21, Chap. 7]. We cannot provide the same level of detail here and try to distill key ideas into the format of this paper. For a more detailed elaboration, we refer the interested reader to [21]. The remainder of the paper is structured as follows: We start with a review of existing metrics in Sect. 2, including the categorization from [3] according to their type and intended area of application. Through this categorization, we select a subset of applicable metrics, provide a closer discussion of these metrics, and ultimately choose two metrics for an evaluation. Next, Sect. 3 states deficiencies for replaceability assessment, proposes a way of dealing with them, and evaluates the selected metrics by means of an exemplary computation. This allows to decide on which metric fits best. Finally, the paper concludes with a summary.

2 Review of Existing Metrics

Similarity metrics, and, therefore, also replaceability metrics, measure the distance between objects [29, 36]. The smaller the distance between the objects is, the more similar and, thus, the more replaceable they are. In our case, the ob-

jects are executable pieces of software. Based on the definition of a similarity metric given in [3, Sect. 2.4], a replaceability metric can formally be defined as follows:

Definition: Replaceability Metric

$$REPL(p^1, p^2) = \frac{1}{1 + dist(p^1, p^2)}, where \tag{1}$$

- $dist : Process \times Process \rightarrow \mathbb{R}_0^+$ is a function that computes the distance between two process models.
- $p^1, p^2 \in P$, where P is the set of all process models and $(P, dist)$ forms the *metric space* [36].

The crucial difference between replaceability metrics lies in their definition of the distance function.

Similarity metrics for process models can further be classified depending on the entities which form the basis of the computation. This results in a classification in terms of *labels*, *structure*, or *behavior* [9].

Label Similarity: Metrics based on label similarity compute the similarity of process models based on the names, i.e., the labels, assigned to their elements. If the labels of two elements found in the models p and p' are identical, these elements are considered to be identical as well. If all elements of p are also found in p', and no more, regardless of the structuring of the process graph, then p and p' are considered to be identical and $REPL(p, p') = 1$. Examples of metrics for label similarity can be found in [1, 6, 18, 25, 34].
The problem with label similarity is that the labels assigned to process elements are normally written in natural language and, hence, they are seldom identical. For instance, the label of an activity in p might be "Check Order", whereas p' contains an activity labeled "Order Checking". Though the labels are not identical, their distance could be considered as small. Certain natural language comparison techniques, such as the string edit distance [22], or semantic techniques that utilize a thesaurus, as for instance found in [6, 10], can be used to improve the similarity computation among labels.

Structural Similarity: Metrics based on structural similarity compare the structure of the process graphs of two models. The smaller the distance between the structure of the graphs, the more similar they are. The distance between graphs can be measured through the graph edit distance of process models [7]. This distance corresponds to the number of insertions, deletions, and substitutions of process elements that are needed to transform the graph of one process model into another. The higher this number is, the greater is the distance. Structural similarity metrics have for instance been defined in [6, 7, 23].

Behavioral Similarity: Metrics based on behavioral similarity focus on the execution behavior of process models [19]. They are often computed based on

the execution traces of process models or the execution dependencies among the activities of two models. First, possible traces or execution dependencies are computed based on the model. Then, these traces or dependencies are compared to the traces or dependencies belonging to another model. The higher the overlap between these sets of traces or dependencies is, the higher is the similarity of the process models. Behavioral similarity metrics can for example be found in [6, 31, 33].
A crucial problem of approaches in this area is the requirement to map the nodes in one process model to one or more nodes in the other model. This is necessary to identify if traces really are similar. To achieve this, approaches for behavioral similarity often make use of approaches for label similarity.

A review of the existing metrics and their classification according to areas of application can be found in [3]. This classification can be used to narrow the set of relevant metrics and to decide which metrics are a potential fit for our use case. In [3], Becker and Laue distinguish seven application areas for similarity metrics: i) The simplification of change in process variants, ii) process merging, iii) facilitation of reuse, iv) management of process model repositories, v) automation of process execution, vi) compliance assurance with normative models, and vii) service discovery. The areas of process execution automation and service discovery are closest to our focus of application. The authors remark that "*automation is usually concerned in SOA applications*" and service discovery is "*closely connected to the goal of automation*" [3, Sect. 4.2]. For these areas of application, the authors recommend behavioral metrics that compute similarity based on the dependencies among the nodes of the process graph in favor of metrics based on label or structural similarity. In particular, these metrics are *dependency graphs* [2] and their improvement in *TAR-similarity* [37], the *string edit distance of sets of traces* [33], *causal behavioral profiles* [31], and *causal footprints* [6]. From this set of metrics, we consider TAR-similarity and causal behavioral profiles to be applicable for a replaceability computation. The reasons for excluding the remaining metrics are explained in the following subsections.

2.1 Direct Precedence Relationships Among Activities

The metrics captured by dependency graphs [2], TAR-similarity [37], and the string edit distance of sets of traces [33], are all based on a similar idea: They compute the similarity of process models by considering direct precedence relationships among the activities in a process model.

In [2], the direct precedence relationships among activities correspond to the so-called *dependency graph* of a process model. To determine process model similarity, first, all direct precedence relationships of two models, i.e., their dependencies graphs, are computed. In a second step, these sets of direct precedence relationships are compared and the higher their overlap is, the more similar the two process models are. More precisely, the distance between the dependency graphs is equal to the number of dependencies that are not present in both of the graphs. In the case of [2], direct precedence relationships among activities are

considered, regardless of gateways that might be placed between two activities. This means that conditional branching or parallelism in a process model is not taken into account. This is a clear drawback of the approach.

An extension of dependency graphs that tries to tackle this issue is formed by TAR-similarity [37]. The term TAR stems from the *transition adjacency relation*, which is a special form of a direct precedence relationship. The TAR does not only consider direct control dependencies among activities, but also takes into account the interleaving of activities that are executed in parallel. This means that if a process model uses inclusive (OR) or parallel (AND) gateways, there is a larger amount of dependencies in the dependency graph, which is called the *TAR set* here. Apart from this, TAR-similarity is computed in the same fashion as dependency graphs, i.e., by comparing the amount of shared adjacency relations of the TAR sets, TAR_1 and TAR_2, of two process models, p^1 and p^2, to all relations. Based on the definition of the similarity metric in [37], the distance function, *dist*, can be defined as follows:

$$dist_{TAR}(p^1, p^2) = \frac{\mid (TAR_1 \cup TAR_2) \mid}{\mid (TAR_1 \cap TAR_2) \mid} - 1 \tag{2}$$

Although TAR-similarity takes gateways into account, it is still limited to direct adjacency relations. An improvement of TAR-similarity that tries to eliminate the requirement of directness can be found in the projected TAR [28]. This extension tries to relax the restriction of direct dependencies, by first computing a projection of the original process model that eliminates so-called silent steps in the model. TAR-similarity is then computed based on the projected model. The problem with the approach presented in [28] is that it cannot automatically be determined which parts of the process model are considered as silent. Here, human judgment is required. For this reason, we omit the projected TAR from further consideration.

Another metric that is quite similar to dependency graphs and TAR-similarity is the string edit distance of sets of traces [33]. This metric is based on the analysis of execution traces. However, as the traces are computed based on the process graph, the difference between sets of traces and a dependency graph is mainly one of terminology. A more notable difference of this metric lies in the fact that it does not necessarily focus on binary, i.e., direct, relations only. Instead, it also takes larger sets of activity sequences into account, which are called *words of length n*, or *n-grams*. An n-gram is a trace of the process model that includes exactly n subsequent activities. To calculate the string edit distance of sets of traces, all possible n-grams for a process model have to be computed. Thereafter, the distance of two process models corresponds to the aggregated string edit distance of all n-grams. Although [33] does not address the computational complexity of this approach, it is clear that the calculation is challenging. As a result, the usage of a high value of n is not feasible in practice and n-grams of length two, called bi-grams, are most frequent. In this case, the string edit distance of sets of traces corresponds to TAR-similarity. For this reason, we only consider TAR-similarity further.

2.2 Causal Footprints

Causal footprints are a behavioral similarity metric proposed in [6]. A causal footprint of a process model corresponds to the set of its activities and the execution dependencies among them. These execution dependencies are not limited to direct precedence relationships. To obtain a causal footprint, a set of look-back links and a set of look-ahead links is computed for every activity in a process model. The set of look-back links of an activity A corresponds to the set of all activities that may precede the execution of A. Similarly, the set of look-ahead links of A corresponds to the set of all activities that may be executed after A has finished. The causal footprint of a process model corresponds to all sets of look-back and look-ahead links of the activities in the model. The sets of look-ahead and look-back links are treated as vectors, and the similarity of two process models is computed through the cosine of their vectors.

Based on their observations in [3], Becker and Laue discourage the usage of causal footprints, due to their computational inefficiency. Despite the usage of the reference implementation of causal footprints and a very moderate test set of only eight small process models, they experienced computation times that were more than five times as large as for any other similarity metric. Hence, we do not consider causal footprints any further.

2.3 Causal Behavioral Profiles

As the name indicates, causal behavioral profiles [31] refer to a behavioral similarity metric. Like the other metrics, its mechanism of computation bases on relations between activities. These relations are not limited to direct precedence relations, as for the metrics discussed in Sect. 2.1. Instead, the metric considers four categories of behavioral relations between activities. Given two activities, A_1 and A_2, these relations are: i) Strict order relation (A_1 is always executed before A_2), ii) co-occurrence relation (if A_1 is executed in a process instance, A_2 must be executed as well, and vice versa), iii) exclusiveness relation (A_1 and A_2 are never executed in the same process instance), and iv) concurrency relation (A_1 may be executed before A_2, but also the other way round). The set of all relations among the activities of a process model is its *behavioral profile*.

The similarity of two process models is computed by comparing their behavioral profiles. More precisely, the amount of shared execution relations among activities is compared to the amount of all execution relations. In this sense, causal behavioral profiles are very similar to TAR-similarity. The main difference between the two lies in what kind of relations among activities are considered. Behavioral profiles are necessarily larger than TAR sets, since they include a relation for every pair of activities in a process model. However, there is an additional notable distinction. To be able to compare the execution relations among activities of the two process models, it is necessary to establish which activities in the two models correspond to each other. Only then, it is possible to determine if two behavioral relations are the same. If it is the case that, for a specific activity, there are no corresponding activities in the partner process

model, all behavioral relations that involve this activity are ignored. As [31] puts it: "*Solely activities that are aligned by the correspondence relation are considered*". This is a significant difference from TAR-similarity, for which such relations are still part of the TAR sets. Furthermore, it implies that causal behavioral profiles heavily depend on a proper correspondence function. Weidlich et al. use a trace equivalence correspondence function based on execution traces and activity labels, but require that the function must be injective. This means it is not applicable if there are activities in the first process model for which no corresponding activity in the second process model can be found.

Table 1. Reviewed Similarity Metrics for Replaceability Evaluation

Metric	Ref.	Object in Focus	Issues
Dependency Graphs	[2]	direct precedence relations	gateways are ignored, superseded by TAR-similarity
TAR-similarity	[37]	direct precedence relations	
Edit Distance of Set of Traces	[33]	n-grams	inefficient for larger n
Causal Footprints	[6]	look-back and look-ahead links	inefficient computation
Causal Behavioral Profiles	[31]	behavioral relations among activities	

The results of the review are summarized in Tab. 1. The table depicts the metrics that seem suitable for our area of application, according to the categorization specified in [3].

3 Metrics Selection

Evaluating the replaceability of executable software is not what the designers of similarity metrics originally had in mind. Due to this, existing similarity metrics share a number of deficiencies for a replaceability evaluation, which we discuss in the following subsection, Sect. 3.1. Based on this discussion, we evaluate the remaining metrics from the previous section using a set of synthetic process models, similar to [3], in Sect. 3.2.

3.1 Deficiencies of Existing Metrics for Replaceability Evaluation

To be applicable in many settings and use cases, all metrics discussed in Sect. 2 are computed on the basis of a formalism, such as Petri nets, or other abstractions of concrete process models. To enable the metrics computation, a process model has to be translated into the formalism. During this translation, language-specific execution semantics of particular language elements, except for control-flow routing constructs, are mostly lost. This is acceptable in general-purpose

application scenarios, in particular those that deal with non-executable and abstract process models anyway. In fact, such scenarios are what most metrics have been designed for. For instance, only three out of the 22 metrics evaluated in [3] are targeted at similarity assessment of executable process models by their authors. This results in several problems when trying to use the metrics for an evaluation of the replaceability of executable software. These problems may render certain metrics unsuitable for this use case.

To understand these issues, it is important to recall the purpose of a replaceability evaluation here: To investigate how well a given executable process model can replace a second model in a given execution environment, because the second one contains language elements that are not supported. As a consequence, the concrete language elements have to be taken into account during the replaceability evaluation. Abstracting from the concrete vocabulary of the language, which is what practically all metrics do, defeats the purpose of the evaluation to begin with. The deficiencies for a replaceability evaluation resulting from this abstraction level can be summarized a follows:

1. *Generality*: In most cases, similarity is computed based on the activities and connectors among them. The vocabulary of a process language may not be limited to these sets of elements. For instance, in the case of BPMN [17], activities and gateways clearly fit to this model. However, it needs to be stated, in what way events fit into the above categories. Moreover, it is seldom made clear how a metric should deal with hierarchical decomposition in a process model.
2. *Node Mapping*: Practically all approaches assume that it is possible to map the nodes, in particular the activities or their execution traces, between two process models, i.e., that it is possible to determine if an activity of process model p_1 *corresponds* to one or more activities of process model p_2. This is referred to as the *matching problem* in [32]. In the definition of most structural or behavioral metrics, this aspect is left open or, if discussed at all, deferred to approaches that determine the correspondence between activities based on their labels. This underspecification is problematic, due to the impact of the node mapping on the similarity metric. As [3, Sect. 7] puts it, "*the quality of the mapping between nodes [...] has a significant contribution to the quality of a similarity measure*". This is even more evident when it comes to executable models. Activity labels on their own are of no importance for the execution semantics of activities. As a consequence, relying on activity labels only for finding corresponding activities is a questionable assumption. Instead, the execution semantics of a concrete activity, as for instance captured in its type (e.g., message sending, script execution, business rule execution, etc.), are a decisive factor. When considering a process language, such as BPMN, a variety of node types with different execution semantics exist. None of the approaches explicitly takes these types into account.

All in all, the issue of generality can be resolved with relative simplicity. In the case of BPMN, as done in [3], events can be included in the replaceability

evaluation by considering them as nodes of a process model in the same fashion as activities. Hierarchical decomposition is more difficult to incorporate. It can be achieved by either treating a *SubProcess* as a single node in the same fashion as tasks and events, or by ignoring the decomposition and embedding the contents of a *SubProcess* into the parent process. The latter is the strategy favored by the metrics we consider for evaluation here [31, 37].

The issue of constructing a node mapping is more critical. By ignoring node types, and, hence, parts of the execution semantics of the process model, a similarity metric risks to rate process models as similar, although they are dissimilar in reality, or vice versa. This can be demonstrated by considering the similarity of the BPMN process models depicted in Fig. 2[1]. These are a reference model, RM, a first variant of the reference model, V1, and a second variant of the reference model, V2. The structure of the process graph is identical in all cases and so are

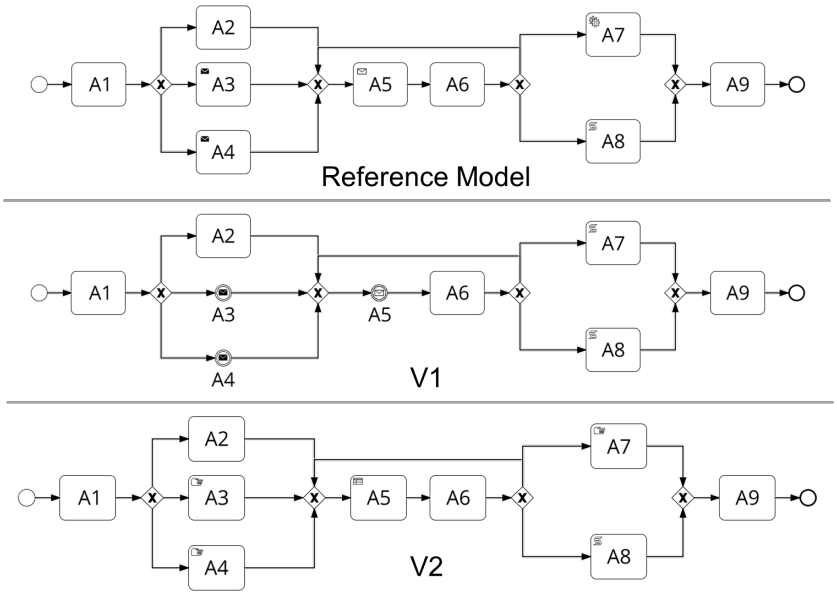

Fig. 2. Process Models for Replaceability Computation

the labels of the nodes. We use identical labels and structure to eliminate any influence of label or structural similarity and to enable an isolated consideration of the node types. When ignoring node types, the dependency graphs and execution traces for the process models, as well as their causal footprints and behavioral profiles are identical. As a result, all process models are considered as identical

[1] The structure of the models is identical to the structure of the reference model from [3]. We adjusted the types of the activities from ordinary tasks to specific BPMN tasks and introduced events in V1.

to each other with the discussed metrics. This is problematic, since two of the models are less similar, when taking the concrete node types into account. At first glance, it may seem that V2 is more similar to the reference model than V1, since V1 uses events, whereas V2 only uses activities. However, the opposite is true: The execution semantics of V1 are identical to RM, although it uses events instead of activities. In V1, *Send-* (A3, A4) and *ReceiveTasks* (A5) are replaced by *Send-* and *ReceiveEvents*. These process elements have the same execution semantics in BPMN. Furthermore, the *ServiceTask* (A7) in RM is replaced by a *ScriptTask*. A script implemented in a programming language can be used to emulate almost any other task, including a *ServiceTask*. Therefore, V1 and the reference model are in fact fully replaceable. In contrast, in V2, activity A5 is not a *ReceiveTask* that blocks until a message is received from an external party, but a *BusinessRuleTask*, which has very different execution semantics. Thus, it cannot be considered as corresponding to activity A5 of the reference model. The same reasoning applies to activities A3, A4, and A7, which are *ManualTasks* in V2, but *Send-* and *ServiceTasks* in the reference model. Similar to [3], the process models depicted in Fig. 2 can be used for evaluating replaceability metrics. A suitable replaceability metric should consider V2 less fit to replace RM than V1 and RM.

To address this issue, we propose a node mapping approach that takes execution semantics of nodes into account to operationalize the metrics for the replaceability evaluation of executable software. [21, Chap. 6] defines a *set of alternatives* for an element of a language in order to measure the adaptability of process models. This set contains all semantically equivalent alternative process elements that can be used to replace a certain element, as described above. Using this set, nodes with similar execution semantics are considered as *corresponding* nodes in the replaceability computation. This notion can easily be combined with approaches for label similarity. For instance, two nodes can be considered as corresponding if their labels are identical *and* their types correspond to each other. Due to limited space, we refer the interested reader to [21] for a formal definition of the respective node mapping function and a mapping of the language elements of BPMN [17].

3.2 Discussion of Metrics Performance

Based on the discussion from the previous section, it is possible to decide between the two remaining metrics, TAR-similarity [37] and causal behavioral profiles [31], through an exemplary computation. As stated in Sect. 3.1, a suitable metric should find V1 to be very similar, if not identical, to the reference model. At the same time, the metric should find V2 to be dissimilar from the reference model. In the following, we discuss the values of these metrics based on the notion of correspondence from the previous section and judge the results, which can be found in Tab. 2. To determine corresponding nodes, we combine label identity and execution semantics.

TAR-similarity: In the case of TAR-similarity, the TAR set of the reference model, TAR_{RM}, resolves to $\{(A1, A2)$, $(A1, A3)$, $(A1, A4)$, $(A2, A5)$, $(A3,$

Table 2. Results of the Metrics Evaluation

Metric	$RM \leftrightarrow V1$	$RM \leftrightarrow V2$
Desired Result	1	< 1
TAR-similarity	1	0.16
Causal Behavioral Profiles	1	1

$A5$), ($A4$, $A5$), ($A5, A6$), ($A6, A7$), ($A6, A8$), ($A7, A9$), ($A8, A9$)}. For V1, the nodes A3, A4, A5, and A7 are events and tasks that differ in their type from their counterparts in the reference model. However, based on the notion of correspondence outlined above, they correspond to the respective tasks in the reference model. Thus, the TAR set of V1, TAR_{V1}, resolves to the same set as for the reference model, TAR_{RM}. Hence, $dist(RM, V1) = 0$ and $REPL(RM, V1) = 1$. Put differently, the TAR-similarity metric rates the reference model and V1 as fully replaceable. This is the desired result.

When considering V2, the tasks A3, A4, A5, and A7 do not correspond to their counterparts in the reference model. Therefore, the TAR set of V2, TAR_{V2} resolves to {($A1, A2$), ($A1, A3_{V2}$), ($A1, A4_{V2}$), ($A2, A5_{V2}$), ($A3_{V2}$, $A5_{V2}$), ($A4_{V2}, A5_{V2}$), ($A5_{V2}$, $A6$), ($A6, A7_{V2}$), ($A6, A8$), ($A7_{V2}, A9$), ($A8$, $A9$)}. Consequently, $TAR_{RM} \cap TAR_{V2}$ is limited to {($A1, A2$), ($A6, A8$), ($A8, A9$)}. Since $\mid TAR_{RM} \cup TAR_{V2} \mid$ resolves to 19, the replaceability metric bears the following value: $REPL(RM, V2) = 1 \ / \ (19 \ / \ 3) \approx 0.16$. The reference model and V2 are not considered as identical.

Summarizing the results, TAR-similarity, when computed based on our our notion of correspondence, passes the evaluation, as it rates RM and V1 as replaceable and RM and V2 as dissimilar.

Causal Behavioral Profiles: Although the source defining causal behavioral profiles [31] uses BPMN process models as example, it states that it is necessary to map the models into a formal representation. This mapping is not clarified in the paper, but deferred to another paper [5]. The applicability of this mapping for BPMN 2.0 process models is questionable, since it refers to an outdated version of BPMN and, therefore, ignores aspects that are imperative to the execution semantics of a BPMN 2.0 process, such as different task types. Nevertheless, using our notion of correspondence, metrics computation for the process models is rather straight-forward. The structure of the three process models is identical, hence, their behavioral profiles are also identical, given they are computed based on node labels or execution traces. As before, the distinguishing difference lies in the nodes that differ between the models: A3, A4, A5, and A7. In the case of the reference model and V1, these nodes are considered to correspond to their counterparts by our notion of correspondence. Therefore, the two process models are considered to be identical and fully replaceable. Again, this is the desired result.

However, the assessment of the replaceability of the reference model and V2 uncovers a crucial problem. For A3, A4, A5, and A7, no corresponding nodes can be found for the reference model in V2 and vice versa. In this

case, as described in Sect. 2.3, all behavioral relations involving these nodes are omitted from the behavioral profiles. This means, replaceability is solely computed by considering the behavioral relations among A1, A2, A6, A8, and A9. These relations are completely identical in both models. Therefore, the result is the same as for the comparison of the reference model and V1: For causal behavioral profiles, $REPL(RM, V2) = 1 = REPL(RM, V1)$ applies, and the two models are considered fully identical. This is not a desirable result, since we expect $REPL(RM, V2) < 1$.
This finding indicates that causal behavioral profiles are not applicable for our use case of computing replaceability. It seems that our notion of node correspondence is too restrictive for a meaningful application of the metric.

The result of the discussion in this section can be expressed as follows: TAR-similarity has been found to be applicable for the evaluation of replaceability. In contrast, causal behavioral profiles have not passed the test, and we cannot recommend this metric for our application scenario.

4 Summary and Conclusion

In this paper, we discussed the suitability of existing metrics for process model similarity, for evaluating the replaceability of executable process models. Essentially, replaceability can be reduced to a problem of similarity. For the computation of this property, a large body of metrics does already exist. A categorization according to the area of application limits the field of promising metrics to dependency graphs, TAR-similarity, the string edit distance of sets of traces, causal footprints, and causal behavioral profiles. The main issue is that the metrics in their current form fail to take node types into account, i.e., the absence of a proper node matching function. When taking node types into account, TAR-similarity [37] seems to be the most promising candidate metric.

The evaluation presented here cannot be considered as a final answer, but rather as a motivating example, and further work is needed. The focus of this work should lie less on the definition of new metrics, but rather on the evaluation of proper node mapping functions. Furthermore, studies that compare metrics would benefit significantly from the existence of sufficiently large corpora of process models that can be used as a benchmark. Such corpora are under development [30], but are still far from being accepted as a reference benchmark.

References

1. Akkiraju, R., Ivan, A.: Discovering Business Process Similarities: An Empirical Study with SAP Best Practice Business Processes. In: 8th International Conference on Service Oriented Computing (ICSOC). pp. 515–526. San Francisco, CA, USA (December 7-10 2010)
2. Bae, J., Caverlee, J., Liu, L., Rouse, W.B.: Process Mining, Discovery, and Integration using Distance Measures. In: International Conference on Web Services. pp. 479–488. Chicago, USA (September 2006)

3. Becker, M., Laue, R.: A Comparative Survey of Business Process Similarity Measures. Computers in Industry 63(2), 148–167 (2012)
4. Bosch, J.: Speed, Data, and Ecosystems: The Future of Software Engineering. IEEE Software 33(1), 82–88 (Jan/Feb 2016)
5. Dijkman, R., Dumas, M., Ouyang, C.: Semantics and Analysis of Buisness Process Models in BPMN. Information and Software Technology 50(12), 1281–1294 (2009)
6. Dijkman, R.M., Dumas, M., van Dongen, B.F., Käärik, R., Mendling, J.: Similarity of Business Process Models: Metrics and Evaluation. Information Systems 36(2), 498–516 (2011)
7. Dijkman, R.M., Dumas, M., García-Bañuelos, L.: Graph Matching Algorithms for Business Process Model Similarity Search. In: Business Process Management Conference. pp. 48–63. Ulm, Germany (September 2009)
8. Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Process-Aware Information Systems: Bridging People and Software Through Process Technology. Wiley (2005), ISBN: 978-0-471-66306-5
9. Dumas, M., García-Bañuelos, L., Dijkman, R.: Similarity Search of Business Process Models. IEEE Data Engineering Bulletin 32(3), 23–28 (2009)
10. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring Similarity between Semantic Business Process Models. In: Asia-Pacific Conference on Conceptual Modelling (APCCM). pp. 71–80. Ballarat, Australia (January/February 2007)
11. Ferme, V., Ivanchikj, A., Pautasso, C.: A Framework for Benchmarking BPMN 2.0 Workflow Management Systems. In: 13th International Conference on Business Process Management (BPM 2015). Innsbruck, Austria (August 2015)
12. Geiger, M., Harrer, S., Lenhard, J., Casar, M., Vorndran, A., Wirtz, G.: BPMN Conformance in Open Source Engines. In: 9th International IEEE Symposium on Service-Oriented System Engineering (SOSE). San Francisco Bay, USA (March/April 2015)
13. Harrer, S.: Process Engine Selection Support. In: OTM Academy. Amantea, Italy (October 2014)
14. Harrer, S., Lenhard, J., Wirtz, G.: Open Source versus Proprietary Software in Service-Orientation: The Case of BPEL Engines. In: 11th International Conference on Service Oriented Computing (ICSOC). pp. 99–113. Berlin, Germany (December 2-5 2013)
15. Hoenisch, P., Schulte, S., Dustdar, S., Venugopal, S.: Self-Adaptive Resource Allocation for Elastic Process Execution. In: IEEE Sixth International Conference on Cloud Computing. pp. 220–227. Santa Clara, CA, USA (June 2013)
16. ISO/IEC: Systems and software engineering – System and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models (2011), 25010:2011
17. ISO/IEC: ISO/IEC 19510:2013 – Information technology - Object Management Group Business Process Model and Notation (November 2013), v2.0.2
18. Klinkmüller, C., Weber, I., Mendling, J., Leopold, H., Ludwig, A.: Increasing Recall of Process Model Matching by Improved Activity Label Matching. In: 11th International Conference on Business Process Management. pp. 211–218. Beijing, China (August 26-30 2013)
19. Kunze, M., Weidlich, M., Weske, M.: Behavioral Similarity – A Proper Metric. In: 9th International Conference on Business Process Management. pp. 166–181. Clermont-Ferrand, France (August, September 2011)
20. Lenhard, J.: Improving Process Portability through Metrics and Continuous Inspection. In: Reichert, M., Oberhauser, R., Grambow, G. (eds.) Advances in Intel-

ligent Process-Aware Information Systems. Springer-Verlag, Germany (2016), to appear
21. Lenhard, J.: Portability of Process-Aware and Service-Oriented Software: Evidence and Metrics. Ph.D. thesis, University of Bamberg, Germany (2016)
22. Levenshtein, V.I.: Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. Soviet Physics Doklady 10(8), 707–710 (1966)
23. Li, C., Reichert, M., Wombacher, A.: On Measuring Process Model Similarity Based on High-Level Change Operations. In: 27th International Conference on Conceptual Modeling. pp. 248–264. Barceclona, Spain (October 2008)
24. Mens, T., Wermelinger, M., Ducasse, S., Demeyer, S., Hischfeld, R., Mehdi, J.: Challenges in Software Evolution. In: 8th International Workshop on Principles of Software Evolution. Lisbon, Portugal (September 2005)
25. Minor, M., Tartakovski, A., Bergmann, R.: Representation and Structure-Based Similarity Assessment for Agile Workflows. In: ICCBR. pp. 224–238. Belfast, Northern Ireland, UK (August 13-16 2007)
26. Newman, S.: Building Microservices. O'Reilly Media (February 2015)
27. Papazoglou, M.P., Georgakopoulos, D.: Service-oriented Computing. Communications of the ACM 46(10), 24–28 (October 2003)
28. Prescher, J., Mendling, J., Weidlich, M.: The Projected TAR and its Application to Conformance Checking. In: Entwicklungsmethoden für Informationssysteme und deren Anwendung (EMISA). pp. 151–164. Vienna, Austria (September 13–14 2012)
29. Santini, S., Jain, R.: Similarity Measures. IEEE Transactions on Pattern Analysis and Machine Intelligence 21(9), 871–883 (1999)
30. Thaler, T., Dadashnia, S., Sonntag, A., Fettke, P., Loos, P.: The IWi Process Model Corpus. Tech. Rep. 199, Publications of the Institute for Information Systems, Saarland Univerity, Saarbrücken, Germany (October 2015)
31. Weidlich, M., Mendling, J., Weske, M.: Efficient Consistency Measurement Based on Behavioral Profiles of Process Models. IEEE Transactions on Software Engineering 37(3), 410–429 (2011)
32. Weidlich, M., Sagi, T., Leopold, H., Gal, A., Mendling, J.: Predicting the Quality of Process Model Matching. In: 11th International Conference on Business Process Management. pp. 203–210. Beijing, China (August 26-30 2013)
33. Wombacher, A., Li, C.: Alternative Approaches for Workflow Similarity. In: 7th International Conference on Services Computing. pp. 337–345. Miami, Florida, USA (July 2010)
34. Yan, Z., Dijkman, R., Grefen, P.: Fast Business Process Similarity Search with Feature-Based Similarity Estimation. In: 18th International Conference on Cooperative Information Systems (CoopIS). pp. 60–77. Crete, Greece (2010)
35. Yan, Z., Dijkman, R., Grefen, P.: Business Process Model Repositories – Framework and Survey. Information and Software Technology 54(4), 380–395 (2012)
36. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: The Metric Space Approach, Advances in Database Systems, vol. 32. Springer (2006), iSBN 978-0-387-29146-8
37. Zha, H., Wang, J., Wen, L., Wang, C., Sun, J.: A workflow net similarity measure based on transition adjacency relations. Computers in Industry 61(5), 463–471 (2010)
38. Zhou, Z., Gaaloul, W., Gao, F., Shu, L., Tata, S.: Assessing the Replaceability of Service Protocols in Mediated Service Interactions. Future Generation Computer Systems 29(1), 287–299 (2013)