Identifying Refactoring Opportunities in Process Model Repositories

Remco Dijkman\textsuperscript{a}, Beat Gfeller\textsuperscript{b}, Jochen Küster\textsuperscript{b}, Hagen Völzer\textsuperscript{b}

\textsuperscript{a}Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands
\textsuperscript{b}IBM Research – Zurich, Säumerstrasse 4, 8803 Rüschlikon, Switzerland

Abstract

Context: In order to ensure high quality of a process model repository, refactoring operations can be applied to correct anti-patterns, such as overlap of process models, inconsistent labeling of activities and overly complex models. However, if a process model collection is created and maintained by different people over a longer period of time, manual detection of such refactoring opportunities becomes difficult, simply due to the number of processes in the repository. Consequently, there is a need for techniques to detect refactoring opportunities automatically.

Objective: This paper proposes a technique for automatically detecting refactoring opportunities.

Method: We developed the technique based on metrics that can be used to measure the consistency of activity labels as well as the extent to which processes overlap and the type of overlap that they have. We evaluated it, by applying it to two large process model repositories.

Results: The evaluation shows that the technique can be used to pinpoint the approximate location of three types of refactoring opportunities with high precision and recall and of one type of refactoring opportunity with high recall, but low precision.

Conclusion: We conclude that the technique presented in this paper can be used in practice to automatically detect a number of anti-patterns that can be corrected by refactoring.

Key words: Business Process Model, Refactoring, Repository

1. Introduction

Today, many organizations maintain repositories that contain hundreds of business process models. For example, the SAP reference model [5], the reference model for Dutch local government [9] and IBM’s insurance architecture (IAA) [15] all contain more than 250 process models.
To promote a joint understanding of the repository and, therewith, re-use and maintainability, the processes in the repository should be modeled as uniformly as possible. In particular, they should use the same terms to describe model elements that have the same meaning and different terms to describe model elements that have a different meaning. To promote maintainability, overlap between processes should be avoided; process parts that appear in multiple business process models should be put into common subprocesses, such that a change to these parts only has to be made in one place. To promote uniformity, the models in the repository should be described at the same level of detail. Refactoring operations can be applied to the process models in a repository to fix problems related to terminology, overlap and level of detail and, therewith, to promote understandability, re-use and maintainability of the models [33]. By refactoring, we mean an operation that changes one or more business process models in a repository, without changing their execution semantics, but improving their understandability, maintainability or reusability.

Refactoring may not only apply to the process models, but also to the real-world processes themselves, saving costs by eliminating redundancies in their implementations.

This paper presents a technique to automatically detect opportunities for applying refactoring operations to fix problems related to terminology, overlap and level of detail. For some problems such as activity naming inconsistencies, simple process matching techniques [1, 7] suffice to detect a refactoring opportunity. However, some refactoring opportunities concern similar process parts, i.e., coherent groups of several related nodes. To find similar process parts, our approach combines process matching techniques with the Refined Process Structure Tree [32], which can be used to group related nodes in a computationally efficient manner.

Identifying similar parts of different processes also has value on its own, without necessarily aiming at refactoring. For example, it can refine the search for similar processes, which is used, e.g., to find re-usable assets that are associated with a similar process, such as a software application or an expert in a role associated with that process. There may be no pair of processes that are similar overall, but there may be processes that have similar parts. Techniques exist to search for similar processes [6]. However, searching similar process parts is in general computationally harder, because it requires that a process is split up into parts and the number of potentially relevant process parts can be exponential in the number of its elements.

This paper has two contributions. Firstly, we show how existing techniques can be combined to efficiently detect many refactoring opportunities in practice. In addition, we present, based on that technique, different metrics to determine the type of problem that the similarity represents, and therewith the type of the refactoring that can be applied. Secondly, we evaluate our technique, by applying it to two real-world reference process model collections. Our evaluation shows that similar process parts occur frequently in those collections and that our similarity metrics can indeed be used for identifying refactoring opportunities.

Our technique focuses only on one aspect of a business process model, viz. on the activities and the groups they form. This restricts our attention to certain important refactorings but we can obtain satisfactory results with a relatively simple technique. Simplicity is not
only helpful to transfer a technique to practical application, but focusing on that one aspect also makes it easier to apply the technique across different modeling languages and styles.

The remainder of this paper is organized as follows. Section 2 defines the refactoring opportunities that we aim to detect. We describe the Refined Process Structure Tree (which originates from [32, 23]) in Section 3 as a basis for determining similarity of process parts. In Section 4, we define different similarity metrics (Section 4.1, which builds on previous work, and Section 4.2, which is new), present a technique for computing similarity of process parts (Section 4.3) and explain its relationship to refactoring opportunities (Section 4.4). Section 5 reports on the results obtained when applying the technique to a repository of process models. Section 6 presents related work, before Section 7 concludes the paper.

2. Refactoring Opportunities in Model Collections

This section defines opportunities for four refactoring operations which improve the quality of a collection of business process models. The focus of this paper is on automatically identifying such refactoring opportunities in a collection. Consequently, we derived the refactoring opportunities by first studying refactoring operations that have been defined in previous work [33, 11, 17]; second, determining the situations in which they can be applied; and, third, selecting the situations that can be detected automatically as refactoring opportunities.

Figure 1 shows the four refactoring opportunities that we have identified in this manner. For each opportunity, it also shows the refactoring operation that can be applied to it in order to create a process model collection that is easier to understand, maintain and re-use.

The first refactoring opportunity is the situation in which there are activities (‘Check claim’ and ‘Verify claim’ in the figure) that are considered to be the same (indicated by the dashed line), but that have different labels. In this situation, a renaming operation should be applied to give the activities uniform labels. The decision as to whether or not the two activities are the same should be made by a human observer based on, for example, brief descriptions of the activities or knowledge about what the activities represent in practice.

The second refactoring opportunity is the situation in which there are process fragments (defined precisely in the next section) that are, as a whole, similar, because the activities and control-flow relations that they are composed of are similar. In this situation the fragment from one of the process models should substitute the other fragment and should be extracted into a common subprocess. Figures 2 a) and c) show two similar process fragments \( F_1 \) and \( F_2 \) which can be extracted into such a common subprocess.

The third refactoring opportunity is the situation in which there are process fragments that are similar, but are composed of both similar activities and activities that appear in one but not in the other fragment. In this case a common subprocess should replace the fragments. This subprocess should be composed of both the similar and the non-similar activities and have the option to skip the non-similar activities, because these activities can be performed in one process but not in the other. Optionally, the ability to perform the non-similar activities can be marked as a configuration option for the process model [26].
Figure 1: Refactoring opportunities and operations

Referring to Fig. 2 a) and b) we now assume that activity names ‘Record claim’ and ‘Recording claim’ as well as ‘Check claim’ and ‘Verify claim’ have been refactored by renaming them such that they have the same label. Then the entire part \(F_{a1}/F_{b1}\) can be extracted from their containing processes with the optional elements ‘Pay out’ and ‘Send letter’. If the activities have to remain different, their names can be adjusted to better reflect the difference and then the parts \(F_{a2}/F_{b2}\) can be extracted with the optional elements ‘Pay out’ and ‘Send letter’.

The fourth refactoring opportunity is the situation in which there are process fragments that are only mildly similar, because they are composed of activities that are similar modulo the business object to which they apply. For example, insurance applications processes for home owner insurance and car insurance are similar, except that they may refer to the different types of insurance in some of their activities. This goes beyond simple name difference, because such activities may also, for example, use different forms and working instructions. In this case a common subprocess should replace the fragments, but activities for which multiple options exist should appear twice, because one activity should be performed in the first process and another in the second process. Optionally, configuration options [26] can indicate that the choice between the two types of activities is configurable and that if one type is chosen, this type applies to all activities in the subprocess. Figures 2 a) and c) show the fragments \(F_{a1}\) and \(F_{c1}\) which are similar modulo the business object and where a common subprocess configured by a business item can be applied.
3. The Refined Process Structure Tree

In order to identify similar parts in two different process models, we first decompose both process models into smaller parts. To do so, we use a parsing technique which is called the Refined Process Structure Tree (RPST) [32, 23], where the smaller parts of the decomposition are henceforth called fragments. Fig. 3 shows an example of such a decomposition. A fragment is a connected subgraph, such that control enters the fragment via a single entry node and leaves the fragment via a single exit node. The RPST defines a unique decomposition, i.e., a hierarchy of fragments, that can be computed in linear time [32]. It can be defined as follows.

Let $G$ be a directed graph with a unique source and a unique sink such that every node is on a path from the source to the sink. For simplicity, we also assume that each node in the graph has either a single incoming or a single outgoing edge. Let $F$ be a connected subgraph of $G$. A node of $F$ is an interior node of $F$ if it is connected only to nodes in $F$, otherwise it is a boundary node of $F$. A boundary node $u$ of $F$ is an entry of $F$ if no incoming edge of $u$ belongs to $F$ or if all outgoing edges of $u$ belong to $F$. A boundary node $v$ of $F$ is an exit of $F$ if no outgoing edge of $v$ belongs to $F$ or if all incoming edges of $v$ belong to $F$. $F$ is a fragment if it has exactly two boundary nodes, one entry and one exit. A fragment is trivial if it only contains a single edge. Note that every singleton edge forms a trivial fragment.
We say that two fragments are *nested* if one is a subgraph of the other, they are *disjoint* if their sets of edges are disjoint. If they are neither nested nor disjoint, we say that they *overlap*. A fragment is said to be *canonical* if it does not overlap with any other fragment. The RPST of $G$ is the set of canonical fragments of $G$. It follows that any two canonical fragments are either nested or disjoint and hence they form a hierarchy. This hierarchy can be shown as a tree, where the parent of a canonical fragment $F$ is the smallest canonical fragment that contains $F$. The root of the tree is the entire graph, the leaves are the trivial fragments, i.e., the edges of the graph. If an activity has a single incoming and a single outgoing edge, then it belongs to a unique fragment. Multiple incoming or outgoing edges usually represent an implicit gateway attached to the beginning or ending of the activity. By making those gateways explicit, we obtain activities with single incoming and outgoing edges.

Fig. 3 shows a tree representation of the RPST for the leftmost process from Fig. 2. Besides the fragments, it also shows the activities of the process model associated with the fragment they belong to. Note that the edge labels that are used in Figs. 2 and 3 were used only for explaining the RPST decomposition in this section. They are henceforth not relevant for the techniques to be presented.

4. Metrics and Computation of Fragment Similarity

In this section, we define various metrics to capture different forms of similarity between process fragments.

The concept of a canonical fragment of the Refined Process Structure Tree defines a notion of a coherent part of a process that can be compared against another fragment, which may be the subject of a refactoring. In this section, we define when two fragments can be considered similar, which may be an indication of a refactoring opportunity. This is based on a matching of their constituent activities, which we explain first.

4.1. Activity Matching

To compute a similarity score for two fragments $F_1, F_2$ of two processes, we first compare the activities they contain, i.e., each activity of $F_1$ is compared with each activity of $F_2$. One of the simplest ways to compare two activities is to compare their names, for example using the string edit distance [18]. The string edit distance is the number of string operations that is necessary to transform one string into the other.
Definition 1 (String edit distance, String edit similarity). Let \( s \) and \( t \) be two strings and let \( |x| \) represent the number of characters in a string \( x \). The string edit distance of \( s \) and \( t \), denoted \( \text{ed}(s, t) \) is the minimal number of atomic string operations needed to transform \( s \) into \( t \) or vice versa. The atomic string operations are: inserting a character, deleting a character or substituting a character for another. The string edit similarity of \( s \) and \( t \), denoted \( \text{Sim}(s, t) \) is:

\[
\text{Sim}(s, t) = 1.0 - \frac{\text{ed}(s, t)}{\max(|s|, |t|)}
\]

For example, the string edit distance between ‘Check claim’ and ‘Verify claim’ from Fig. 2 is six; delete ‘C’, substitute ‘h’ with ‘V’, substitute ‘ck’ with ‘ri’ and insert ‘fy’. Consequently, the string edit similarity is \( 1.0 - \frac{6}{12} \). Techniques for computing the string edit distance are well known (e.g.: [18]).

Alternative techniques for computing the string similarity score exist [8, 30]. We use string edit similarity in this paper, because it is easy to explain and because it produces results that are not much worse than more complex techniques for the type of process model collections that we use for evaluation [8, 30]. In principle, the string edit similarity can be easily replaced in our method, to use a more complex one that may perform better, e.g. taking semantic ontologies into account. However, our evaluation does not explore this possibility.

In order to have a binary notion of similarity, i.e., two strings are either similar or not similar, we introduce a threshold value \( \theta, 0 \leq \theta \leq 1 \) such that two strings are considered to be similar if \( \text{Sim}(s, t) \geq \theta \).

An activity of \( F_1 \) can be matched with an activity of \( F_2 \) if they are similar. Let \( A_i \) be the set of activities of \( F_i \) for \( i = 1, 2 \). An activity matching is a relation \( M \subseteq A_1 \times A_2 \) such that each activity in \( A_1 \) is related to at most one activity in \( A_2 \) and each activity in \( A_2 \) is related to at most one activity in \( A_1 \). Hence, we assume in this approach that one activity in one fragment cannot represent multiple activities in the other fragment. This assumption simplifies the computation but might decrease the quality of the results in some use cases. However, we do not expect a substantial negative impact because we are only interested here in a quantitative similarity measure and not in an exact matching of similar activities (such as in [7]).

An activity matching is optimal if the sum of the similarity scores of the matched pairs is maximal. The optimal matching can be computed in cubic time using a maximum graph matching algorithm [12] or approximated within a factor of two using a greedy algorithm [6]. For example, in Fig. 2, ‘Record claim’ can be matched with ‘Recording claim’ and ‘Check claim’ with ‘Verify claim’. This matching has a similarity score of \( 0.8 + 0.5 \) (for only these two activities). Alternatively, ‘Record claim’ can be matched with ‘Verify claim’ and ‘Check claim’ with ‘Recording claim’, which has a similarity score of \( 0.5 + 0.47 \). Hence, the first matching would be preferred over the second matching with respect to these two activities.

Note that multiple optimal activity matchings can exist, because multiple activity matchings have the same, maximal, similarity score. If multiple optimal activity matchings exist, we select one randomly.
4.2. Fragment Similarity Metrics

We consider two aspects for determining the similarity between two fragments:

- How many of the activities of one fragment have a matching activity in the other fragment? (*matching completeness*)
- How similar are the activities that are matched? (*matching quality*).

Each of the aspects can be captured in a similarity metric and both can be aggregated in a common metric. We formalize this as follows.

**Definition 2 (Completeness similarity).** Let $F_1$ and $F_2$ be two fragments of two processes and let $A_1$ and $A_2$ be their respective sets of activities. Furthermore, let $M \subseteq A_1 \times A_2$ be an optimal activity matching. The **completeness similarity** for $F_1$ and $F_2$ is defined as:

$$\text{SimC}(F_1, F_2) = \frac{|M|}{\max(|A_1|, |A_2|)}$$

The following metric captures how well activities match by taking the similarity scores into consideration, but only for those pairs that are actually matched.

**Definition 3 (Quality similarity).** Let $F_1$, $F_2$, and $M$ be defined as in Def. 2. Assume further that $|M| \geq 1$. Let $\text{name}(a)$ denote the name of a activity $a$. The **quality similarity** for $F_1$ and $F_2$ is defined as:

$$\text{SimQ}(F_1, F_2) = \frac{\sum_{(t_1, t_2) \in M} \text{Sim}(\text{name}(t_1), \text{name}(t_2))}{|M|}$$

We can aggregate the two metrics above by using their weighted average. A weight indicates the relative importance of a particular metric, which can be assigned by the process analyst.

**Definition 4 (Combined fragment similarity).** Let $F_1$ and $F_2$ be defined as in Def. 2 and let $0 \leq \gamma \leq 1$. We define:

$$\text{SimA}(F_1, F_2) = \gamma \cdot \text{SimC}(F_1, F_2) + (1 - \gamma) \cdot \text{SimQ}(F_1, F_2)$$

Similar to activity similarity discussed in Sect. 4.1, we can define a binary fragment value by introducing a threshold value $\tau$ and say that a fragment pair $(F_1, F_2)$ is **similar** if $\text{SimA}(F_1, F_2) \geq \tau$.

In the following, we first discuss how to compute fragment similarity before we elaborate on how the metrics can be used for identifying refactoring opportunities.
4.3. Computation of Fragment Similarity

Suppose we want to find all similar pairs of fragments within a given repository of processes. In the following, we explain how to obtain all these similar fragments. An overview of the necessary steps is given in Algorithm 1.

First, the refined process structure tree is computed for each process in the repository, which gives rise to a collection of process fragments. Then, we compute a full fragment similarity matrix, i.e., for each pair of fragments from this fragment collection, we determine the similarity value for that pair. If the similarity value of a fragment pair is below the preset threshold $\tau$, we delete it. The remaining pairs of similar fragments could be presented to the user in decreasing order of their similarity value. However, there may be many pairs which are redundant in the following sense and which can be filtered out:

If a fragment $X$ of a process $p$ is similar to a fragment $Y$ of a process $q$ then it is likely that the parent $X'$ in the RPST of $X$ is also similar to $Y$. It is then not necessary to display both $(X,Y)$ and $(X',Y)$ to the user to point out the similarity. Instead, we can either display the maximal pair in the RPST hierarchy, i.e., $(X',Y)$ in this case, provided that its similarity is still above the threshold, or we display the pair with the higher similarity value.

We formalize this as follows. We say that $(X',Y')$ subsumes $(X,Y)$ if $X \subseteq X', Y \subseteq Y'$ and $\text{SimA}(X',Y') > \tau$. We say $(X',Y')$ supersedes $(X,Y)$ if $X \mid X', Y \mid Y'$ and $\text{SimA}(X',Y') > \text{SimA}(X,Y)$ where $X \mid X'$ (X is compatible with $X'$) is true whenever $X \subseteq X'$ or $X' \subseteq X$. Thus we clean the matrix from similarity pairs that are not subsumed (resp. superseded) by any other similarity pair. As an example, consider again Fig. 2. The pair $(F_{a2}, F_{c2})$ is subsumed by and supersedes the pair $(F_{a1}, F_{c3})$.

**Algorithm 1** Algorithm to compute the set of similar fragment pairs for a given set $P$ of processes, a fragment similarity threshold $\tau$ and a weight $\gamma$ for SimA.

```plaintext
for each $p \in P$ do
    Compute the RPST of $p$
end for
for each process pair $p_1, p_2 \in P$ and each fragment $F_1$ of $p_1$ and $F_2$ of $p_2$ do
    compute the similarity $\text{SimA}(F_1, F_2)$ using weight $\gamma$
end for
Delete all similarity pairs $(F_1, F_2)$ where $\text{SimA}(F_1, F_2) < \tau$
Delete all similarity pairs that are subsumed (resp. superseded) by another similarity pair
Output all remaining similarity pairs ordered by similarity value
```

4.4. Similarity Metrics and Refactoring Opportunities

Different ‘completeness’ and ‘quality’ values may indicate different refactoring opportunities. Here we formulate our expectations about how the completeness and quality may differ for the different refactoring opportunities that we address in this paper (see Fig. 1):

- (Rename activities) This refactoring opportunity applies to activities that have similar, but not identical names, because in that case it is likely that their names should be
reconsidered in order to either unify them or clearly distinguish them. Consequently, we expect that this refactoring opportunity can be identified by high similarity of activity labels.

- (Extract subprocess) This refactoring opportunity applies to fragments that are highly similar, because in that case it is likely that they represent the same process part, which can be extracted into a common subprocess. Consequently, we expect that this refactoring opportunity can be identified by both high quality and high completeness of fragment similarity. As we only look at activity names in our metrics, the two fragments can still differ in the business items and control flow used. In the following, we assume that if different business items are used, then also the activity naming will be different. Strictly speaking a subprocess can only be extracted in case the fragments have equivalent behavior. To determine this, existing techniques for determining behavioral equivalence can be used [31].

- (Extract subprocess with optional elements) This refactoring opportunity applies to fragments that are similar in part, but also have model elements in one fragment that the other fragment does not have. Consequently, we expect that this refactoring opportunity can be identified by high quality, but low completeness.
• (Extract subprocess with varying business items) This refactoring opportunity applies to fragments that are similar, but that apply to different business items. Fig. 4 shows an example of such a process fragment. Both process fragments deal with insurance claims, but one deals with homeowner insurance claims, while the other deals with car insurance claims. We assume here that activity naming refers to the business items that are used, which then leads to the situation that there is a slight difference in the activity naming. Consequently, we expect that this refactoring opportunity can be identified by high completeness but medium quality fragment similarities. In future work we can further refine the detection of this refactoring opportunity, by using information that is available in modeling tools about the business items that are used in an activity.

In the next section we will test these expectations and we will show to which extent the quality and completeness measures can be used to automatically identify the different types of refactoring opportunities.

5. Evaluation

We evaluated the capability of our technique to find refactoring opportunities in three steps. In step one, we applied the technique to merely identify similar fragments, without classifying them as specific refactoring opportunities. The goal of this step is twofold. Firstly, we use this step to evaluate how well our technique is capable of finding similar fragments, because finding similar fragments is the basis for finding refactoring opportunities. Secondly, we use this step to determine good starting values for the parameters $\gamma$, $\theta$, and $\tau$ that we introduced in this paper and whose values will influence the quality of the results. In step two, we investigate in more detail fragment similarities that are identified by the technique. The goal of this step is to study which properties of similar fragments could be used to classify them as specific refactoring opportunities. Although the results of this step are only presented here, they influenced certain choices that were presented earlier in the paper. In particular, they also lead to the expectations that are specified in Section 4.4. In step three, we investigate how well our technique is capable of classifying similar fragments as refactoring opportunities, using the properties that are identified in the second step.

We evaluated our technique by applying it to find refactoring opportunities in two reference models; the IBM Insurance Application Architecture model [15] (IAA) and the SAP Reference Model [5] (SAP-RM).

Both of these models contain best-practice business process models that were developed by consultants, after studying the process domains in a number of organizations. Both collections of models are typical reference models and, as we will show in the first evaluation, they have similar properties for the purpose of our evaluation. Therefore, we expect that the conclusions that we draw based on them are representative for collections of reference process models in general. One property of the reference models is that their node labels are aligned (i.e.: the consultants who developed the collection ensured that similar nodes have similar labels). This is an important property that strongly affects our evaluation results.
We expect that our conclusions extend not only to other collections of reference models, but to all collections of business process models with the property that labels of similar nodes are themselves similar. In process repositories more inhomogeneous than those we used for the evaluation, we would expect a somewhat deteriorated performance of our method.

Note that we did not verify the opportunities by trying to refactor the process models, but by judgement of a human observer. Thus, in principle there could be cases where the human classifies a refactoring opportunity found by our algorithm as valid, even though it would turn out not to be the case when trying to do the refactoring. However, we believe that if the node labels in the process repository are aligned, such cases are relatively rare.

The IAA reference library we used contains 284 process models. On average, a model in this library contains seven to eight activities, and the average length of an activity label is about four words. The SAP-RM contains 604 process model. On average, a model in this library contains four to five activities, and the average length of an activity label is three to four words.

This sections presents four evaluations. As a first evaluation, we determined to which extent fragment similarity occurs in the reference model repositories. As a second evaluation, we determined which levels of ‘completeness’ and ‘quality’ determine which type of refactoring opportunity. As a third evaluation, we determined how well refactoring opportunities can be identified using these levels of ‘completeness’ and ‘quality’. As a fourth evaluation, we qualitatively compared and investigated the values that are returned by the different metrics introduced in Section 4.

5.1. Similarity in Reference Models

We applied our technique to both reference models to determine to which extent fragment similarity occurs within these collections. We expect fragment similarity to occur frequently and on various levels of granularity, because both collections contain redundancy by construction. Reasons for redundancy include the following. Firstly, similar activities or collections of activities are used in different processes. For example, in many business process models, a decision must be validated by another employee or a superior, leading to the introduction of an activity for that purpose. Also, in a few business process models, the client has the option to appeal a decision. The appeal is a collection of steps. Secondly, business processes may be similar to a large extent and differ only with respect to some details. For example, the business processes for sales of a home insurance and sales of a car insurance differ mainly with respect to the type of insurance that is sold.

To use the techniques, we must first establish values $\gamma$ and $\tau$ for which appropriate overlapping fragments are returned (see Definition 4), because if we set these values wrong, fragments may be returned that are not perceived to be similar by a human observer or fragment similarities may be missed. For example, if we set $\tau = 0$ then all fragment pairs will be considered similar, which will lead to too many fragment pairs that are considered irrelevant by a human observer.

We established appropriate values for $\gamma$ and $\tau$ as follows. For matching activities, we used a greedy matching algorithm, and used $\theta = 0$ (see Section 4.1). We use $\theta = 0$ to make our conclusions independent of the specific value of $\theta$, such that as little parameterization
as possible is needed to use the algorithm for an arbitrary collection of process models. A value of \( \theta \) allows all matches between activities to be considered as potential matches. We selected 50 pairs of process models from the SAP-RM that were evenly distributed over the spectrum ‘identical’ to ‘completely dissimilar’ (i.e. using Definition 4 and \( \gamma = 0.5 \), we selected an equal number of pairs with a rounded similarity of 0, a rounded similarity of 0.1, \ldots and a rounded similarity of 1.0). Subsequently, three colleagues manually determined pairs of similar activities for each of the 50 pairs of process models. We consider a pair of similar activities in case two of the three colleagues considered the pair to be similar. The result of this step is a set of pairs of activities that contains manually selected pairs of similar activities. We then automatically determined pairs of similar activities (i.e.: similar fragments) between the two processes using Definition 4, experimenting with different values of \( \gamma \) and \( \tau \) to determine which values yield the best results in terms of precision and recall (i.e.: that result in the lowest number of false negatives and false positives). In particular, we determined the highest f-score. Precision, recall and f-score are defined as follows.

**Definition 5 (Precision, Recall, F-Score).** Let \( A \) be the set of all possible activities. Furthermore, let \( S_{man} \subseteq P(A) \times P(A) \) be the set of manually determined pairs of similar activities. We also say that these pairs are the relevant pairs, because they are considered to be relevant by the human observer. Let \( S_{aut} \subseteq P(A) \times P(A) \) be the set of automatically determined pairs of similar activities for some values of \( \gamma \) and \( \tau \). We say that an automatically determined pair of similar activities, \((A_1, A_2) \in S_{aut}\), is also a relevant pair in case there exists a manually determined pair of similar activities, \((A'_1, A'_2) \in S_{man}\), that overlaps with it in both process models. Formally:

\[
(A_1, A_2) \text{ Relevant to } (A'_1, A'_2) \iff (A_1 \cap A'_1 \neq \emptyset) \land (A_2 \cap A'_2 \neq \emptyset)
\]

We now define precision and recall as usual. *Precision* is the fraction of automatically determined pairs that is also relevant. *Recall* is the fraction of relevant pairs that is also automatically determined.

\[
\text{Precision} = \frac{|\{P \in S_{aut} | \exists P' \in S_{man} : P \text{ Relevant } P'\}|}{|S_{aut}|}
\]

\[
\text{Recall} = \frac{|\{P' \in S_{man} | \exists P \in S_{aut} : P \text{ Relevant } P'\}|}{|S_{man}|}
\]

\[
F\text{-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\]

Specifically, the definition of relevance requires that, for an automatically determined pair of activities to be relevant, it has at least one overlapping activity with a manually determined pair of activities in each process. This implies a loose notion of relevance that should be interpreted as follows. We are interested in finding refactoring opportunities (or overlap) for a user, but we accept that usually the user will have to manually adapt the refactoring opportunities that are found, by including or removing activities. The definition of relevance
implies that each automatically determined refactoring opportunity at least pinpoints an approximate location of a refactoring opportunity in a pair of processes. In our evaluation the precision and recall scores indicate the accuracy with which the approximate location can be pinpointed.

We obtained the best result for $\gamma = 0.5$ and $\tau = 0.7$. For these parameter settings, the precision was 1.0 and the recall 0.98.

For the IAA library, we used a very similar approach, except that we manually determined the pairs of similar activities by one person only, using again $\gamma = 0.5$ and $\tau = 0.7$.

Using these settings, we determined the number of similar fragments in the reference model and the sizes of these fragments. We used the supersedes relation (see Section 4.3) which typically favors smaller fragments over larger ones. We obtained a precision of 0.72 and a recall of 0.94, which gives an f-score of 0.82. The relatively low precision is mostly due to small spurious fragments that the automatic method found in processes with low similarity (a completeness score below 0.35). Most of these 'wrong' fragments contain matched activities with low similarity. However, since no activity matching threshold was used (i.e.: $\theta = 0$), their completeness is very high (sometimes 1.0), so their combined score is still above the 0.7 threshold. Using an activity matching threshold above 0 would remove these fragments, because then the completeness score would drop. Therefore, we use $\theta = 0.5$ in the remainder of the experiments. This threshold will exclude matches between activities that share less than half the number of characters in their labels, thus we consider this threshold to be both reasonable and not a risk to the generalizability of the results.

Fig. 5 shows the result of applying our technique to find similarities in the two reference models as a size histogram, where the number of occurrences of similar fragment pairs of a given size is indicated. The size of the fragment pair is presented as the size of the largest fragment in the pair. The size histogram shows that we find many pairs of similar fragments in our reference models. In particular there exist many (thousands) small similar fragments. These small fragments are less relevant for the purpose of refactoring, because refactoring overlapping fragments mainly involves extracting the common part of the fragments into a common subprocess and extracting a subprocess for two elements does not improve the clarity of the process models very much. However, there are also many similar fragment pairs of larger sizes, for which refactoring could be helpful to improve the clarity and maintainability of the business process models.

Based on the histogram from Fig. 5, we conclude that similarity and therewith overlap of process model fragments indeed occurs frequently in reference models. We also conclude, based on the high precision and recall with which overlap between reference models is determined, that our technique is useful for finding intuitively similar fragments.

5.2. Quality and Completeness Scores of Refactoring Opportunities

In this part of the evaluation, we test the expectations that are formulated in Section 4.4 for identifying refactoring opportunities in similar fragments. For this purpose, we chose a setup with $\theta = 0.5$, $\gamma = 0.5$ and $\tau = 0.7$ and then computed the three similarity metrics in the IAA reference model. We then manually inspected 59 similar fragment pairs and determined the refactoring opportunity that each pair represented.
For each pair, we then determined the quality and completeness similarity scores. The results are shown in Fig. 6. The expectations that we want to test are: for ‘extract subprocess’ refactoring opportunities completeness is high and quality is high; for ‘extract subprocess with optional elements’ opportunities completeness is lower and quality is high and for ‘extract subprocess with varying business items’ opportunities completeness is high and quality is medium. The results show that on average:

- for ‘extract subprocess’ completeness is 1.0 and quality is 0.90,
- for ‘extract subprocess with optional elements’ completeness is 0.64 and quality is 0.92,
- for ‘extract subprocess with varying business items’ completeness is 1.0 and quality is 0.89.

These results support our expectations: We can use the values for completeness and quality of fragment similarity to determine the type of refactoring opportunity represented by two overlapping fragments. The results also show that while there indeed exists the expected difference in quality between ‘extract subprocess with optional elements’ and ‘extract subprocess with varying business items’, this difference is very small, potentially leading to complications when distinguishing these types of refactoring opportunities.

In the next subsection we will investigate in detail to which extent we can distinguish the different refactoring opportunities correctly.
5.3. Evaluation of Classification of Refactoring Opportunities

In this section, we evaluate to which extent our metrics can be used to classify refactoring opportunities correctly as a ‘rename activities’, an ‘extract subprocess’, an ‘extract subprocess with optional elements’ or an ‘extract subprocess with varying business items’ opportunity. We perform this evaluation by automatically classifying the similar fragments that were found in the 50 process model pairs from the SAP-RM as described in Subsection 5.1. To do the automatic classification, we use the following thresholds to determine what is considered to be ‘high’, ‘medium’ or ‘low’ quality and completeness in Section 4.4:

- $\sigma$, such that similar fragments with a completeness (Definition 3) greater than or equal to $\sigma$ are considered to be highly complete;
- $\upsilon_{\text{high}}$, such that similar fragments with a quality (Definition 2) greater than or equal to $\upsilon_{\text{high}}$ are considered to have a high quality similarity; and
- $\upsilon_{\text{med}}$, such that similar fragments with a quality greater than or equal to $\upsilon_{\text{med}}$, but less than $\upsilon_{\text{high}}$, are considered to have a medium quality similarity.

We compare the automatic classification to a manual classification that is made by the same three colleagues who determined the similar fragments in Subsection 5.1.

Table 1 shows the results of using the metrics to detect refactoring opportunities. For each type of refactoring opportunity it shows the number of times it appears in the collection.
Table 1: Refactoring opportunities in the evaluation set

<table>
<thead>
<tr>
<th>refactoring opportunity</th>
<th>in collection</th>
<th>precision</th>
<th>recall</th>
<th>f-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>rename activities</td>
<td>15</td>
<td>1.0</td>
<td>0.87</td>
<td>0.93</td>
</tr>
<tr>
<td>extract subprocess</td>
<td>20</td>
<td>0.86</td>
<td>0.60</td>
<td>0.71</td>
</tr>
<tr>
<td>extract subprocess with optional elements</td>
<td>18</td>
<td>0.75</td>
<td>0.83</td>
<td>0.79</td>
</tr>
<tr>
<td>extract subprocess with varying business items</td>
<td>3</td>
<td>0.38</td>
<td>1.00</td>
<td>0.55</td>
</tr>
</tbody>
</table>

according to the human observer. It then presents the accuracy of the results produced using the metrics in terms of precision, recall and f-score.

Table 1 shows that the ‘rename activity’ opportunities can be detected with a high level of accuracy. ‘Extract subprocess’ and ‘extract subprocess with optional elements’ opportunities can be detected with an acceptable level of accuracy. ‘Extract subprocess with varying business items’ can be detected with a very low level of accuracy; only 38% of the refactoring opportunities detected in this class are also identified as a refactoring opportunity by the human observer. This last refactoring opportunity is based on the assumption that somehow the business item to which an activity applies is captured in the label of the activity. We verified that this assumption was true for the three cases that we identified, i.e.: we made sure that we validated the ability to detect the refactoring opportunity and not the assumption. Nonetheless, it may be possible to improve the detection of this refactoring opportunities by explicitly considering the information aspect, rather than relying on the activity labels to capture the business item to which the activity applies.

We determined the thresholds for which the f-scores for detection of refactoring opportunities of each type are the highest (i.e. the scores from Table 1). Table 2 shows the thresholds for which this is the case. For all experiments, we used $\gamma = 0.5$. In principle, we also used $\theta = 0.5$. However, the results that we got with this value forced us to use other values. In particular, using $\theta = 0.5$ for the ‘rename activities’ opportunities gave us a very low precision (0.03 to be precise). Further exploration of different values of $\theta$ showed that using $\theta = 0.9$ gives the best results in terms of f-score. Using $\theta = 0.5$ for ‘extract subprocess with varying business items’ opportunities gave us a recall of 0. Further exploration of different values of $\theta$ showed that using $\theta = 0$ gives the best results in terms of f-score. For ‘rename activities’ opportunities, no thresholds other than $\theta$ are necessary, because these opportunities are detected by optimizing the total similarity score of all matched activities. ‘Extract subprocess’ and ‘extract subprocess with optional elements’ opportunities can be detected with the same thresholds. ‘Extract subprocess with varying business items’ opportunities require a different threshold.

These results lead to the conclusion that three out of four refactoring opportunities that are considered in this paper can be detected with an acceptable level of accuracy, using the metrics that are provided for this purpose. In particular ‘rename activities’ can be detected with a high level of accuracy. For the other refactoring opportunities, improvement is possible. The detection of ‘extract subprocess with varying business items’ opportunity has an unacceptably low accuracy. However, note that we had a comparatively small number of such refactoring opportunities in our sample.
5.4. Comparison of Fragment Similarity Measures

In Section 4, we introduced different similarity measures: quality, completeness, and a combined measure. To see how these measures relate to each other, and to relate them to the size of the fragments (measured as the number of matched activities between the two fragments), we created a scatterplot matrix containing all fragments resulting from running our similarity search on the IAA library. This visualization consists of a matrix whose elements are scatterplots corresponding to two particular measures. In each scatterplot, each similar fragment found is represented as a point, whose coordinates are defined by the two values of the two measures that this particular fragment has. The resulting plot is shown in Fig. 7. One can see that the completeness score is equal to 1.0 for most fragment pairs. This might be an intrinsic property of the IAA reference library. On the other hand, we suspect that this is at least partly due to the fact that we used an activity similarity threshold of 0 (which allows activities to be matched even when their labels are quite similar). Also, observe that the quality score of a fragment is completely independent of its size, as is expected by its definition (this is apparent in the lower left scatterplot of the matrix). One can also see that completeness and quality are very different (i.e., uncorrelated) measures, and therefore complement each other well.

6. Related Work

Refactoring operations that can be applied to process models are defined in three papers [33, 17, 11]. The refactoring operations that we propose are taken from these papers. However, to the best of our knowledge, this is the first paper that presents techniques for automatically identifying opportunities to apply these operations. The existing papers on refactoring operations propose more operations than just the ones that we have used. We focus on operations that can be applied directly to improve the quality of a collection of process models. Other operations, such as moving, inserting and deleting a process model fragment are out of the scope of this paper, because they are used between different versions of the same process model.

We search for refactoring opportunities by applying techniques to find matches between process model elements. There exist three research efforts that develop techniques for matching process model elements [1, 7, 34]. Brockmans et al. [1] match process model elements using both structural properties of the process model and semantic annotations of the business process elements. Dijkman et al. [7] use structural properties of the business processes

<table>
<thead>
<tr>
<th>refactoring opportunity</th>
<th>( \theta )</th>
<th>( \tau )</th>
<th>( \sigma )</th>
<th>( \upsilon_{\text{high}} )</th>
<th>( \upsilon_{\text{med}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>rename activities</td>
<td>0.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>extract subprocess</td>
<td>0.5</td>
<td>0.83</td>
<td>0.91</td>
<td>0.60</td>
<td>0.20</td>
</tr>
<tr>
<td>extract subprocess with optional elements</td>
<td>0.5</td>
<td>0.83</td>
<td>0.91</td>
<td>0.60</td>
<td>0.20</td>
</tr>
<tr>
<td>extract subprocess with varying business items</td>
<td>0.0</td>
<td>0.81</td>
<td>0.71</td>
<td>0.98</td>
<td>0.90</td>
</tr>
</tbody>
</table>
Figure 7: A scatterplot matrix that shows correlations and differences between the measures when applied to the IAA library.

and labels of the business process elements. Weidlich et al. [34] present a technical framework for matching business process model elements. In particular they address the issue of the increase in computational complexity when matching an element in one process model to multiple elements in another process model.

Techniques for matching process model elements rely on techniques for measuring similarity of business process models (and their elements). A number of techniques for measuring business process similarity exists [4, 22, 30, 19, 35, 20, 21, 10]. They are compared in [6]. We go beyond these techniques by proposing the similarity of process parts.

The problem of matching business process elements is strongly related to matching database schema elements, which is an extensively studied area of research [24, 27]. However, although parts of database schema matching techniques can be used in the area of business process matching, the techniques must be adapted to be useful in this area. Matching techniques from the database schema matching area rely on edges having labels, which
are often absent in business process models. In addition to that database schemas have a mesh-like structure, while business process models have a linear flow with a start and an end. We experimented with one particular database schema matching approach called similarity flooding and reported on the results in [8].

The problem of matching business process element using the RPST is also related to techniques for XML document similarity and matching (of which an overview is presented in [29]), because both the RPST and XML documents have a tree-based structure that can be used to determine their similarity. The main difference between XML document similarity and RPST similarity is that within the RPST control-flow relations between process elements are preserved and can be used for similarity measurement. Moreover, the RPST itself is constructed based on the control-flow relations between process model elements, while XML document similarity is based purely on the tree structure that is explicit in the document itself.

Both process matching and database schema matching are derived from the general area of graph matching, which is used in various areas, e.g., molecule structure comparison, computer vision or pattern recognition [2, 3]. Although one can formulate constraints on graphs that allow to compute the graph matching in polynomial time [14, 16], many forms of graph matching are known to be NP-complete. This is true in particular for the computation of the similarity between graphs and for computing the maximum common subgraph (which could be used for detecting overlap between business processes). There exist numerous heuristic techniques for finding different graph matchings (see e.g. [28, 25, 13]), but few provide approximation guarantees for the quality of the computed solution.

7. Conclusion

In this paper, we presented a technique for detecting refactoring opportunities in process model repositories. The technique works by first computing activity similarity and then computing three similarity scores for fragment pairs of process models. Using these similarity scores, four different kinds of refactoring opportunities can be systematically identified.

We evaluated our technique by applying it for finding similar fragment pairs in two reference model libraries and for identifying refactoring opportunities. Our evaluation shows that similar fragments occur frequently in practice. It further shows that our metrics can be used for systematically identifying refactoring opportunities. We have also shown that three out of four refactoring opportunities can be detected with a high level of accuracy.

Our present technique cannot be used (nor aims to) automatically identify all opportunities for applying refactoring operations. When discounting for overlap between the papers, Weber and Reichert [33], Küster, Koehler and Ryndina [17], and Fettke and Loos [11] present seven refactoring operations and for each of these operations there may exist multiple opportunities to apply them. Our technique can currently be used to identify four refactoring opportunities of these. This leaves many opportunities for future work to extend the possibilities for detecting refactoring opportunities by extending and enriching our present technique. In particular, the refactoring opportunities that we can currently address
all focus on the control-flow aspect of business process modeling, while other aspects, such as the resource or the data aspect, can be refactored or play a role in refactoring as well.

In addition to that, the detection and evaluation of these four refactoring opportunities, including subprocesses ‘with optional elements’ and with ‘varying business items’ can be further improved by taking into account further features of the process model (e.g., the business item itself). In addition, we intend to provide tool support for automating the refactoring operations themselves.

Acknowledgements

We would like to thank Masud Fazal-Baqiae, Alexandru Marinescu and Viktor Mueller for help in validating our technique.

References


