Meronymy-based Aggregation of Activities in Business Process Models

Sergey Smirnov[1], Remco Dijkman[2], Jan Mendling[3], and Mathias Weske[1]

[1] Hasso Plattner Institute, University of Potsdam, Germany
{sergey.smirnov,mathias.weske}@hpi.uni-potsdam.de
[2] Eindhoven University of Technology, The Netherlands
r.m.dijkman@tue.nl
[3] Humboldt-Universität zu Berlin, Germany
jan.mendling@wiwi.hu-berlin.de

Abstract. As business process management is increasingly applied in practice, more companies document their operations in the form of process models. Since users require descriptions of one process on various levels of detail, there are often multiple models created for the same process. Business process model abstraction emerged as a technique reducing the number of models to be stored: given a detailed process model, business process model abstraction delivers abstract representations for the same process. A key problem in many abstraction scenarios is the transition from detailed activities in the initial model to coarse-grained activities in the abstract model. This transition is realized by an aggregation operation clustering multiple activities to a single one. So far, humans decide on how to aggregate, which is expensive. This paper presents a semi-automated approach to activity aggregation that reduces the human effort significantly. The approach takes advantage of an activity meronymy relation, i.e., part-of relation defined between activities. The approach is semi-automated, as it proposes sets of meaningful aggregations, while the user still decides. The approach is evaluated by a real-world use case.

1 Introduction

As organizations increasingly work in a process-oriented manner, they create and maintain a growing number of business process models. Often several hundred or even thousand of process models are stored in a company's repository. There are two reasons contributing to this growth. On the one hand, modeling initiatives formalize a multitude of operational processes; on the other hand, one process is often described from different perspectives and at various levels of detail. This increasing amount of models poses a considerable challenge to repository management. The BPM community has targeted this type of problems with, for example, techniques to efficiently deal with process model variety [16, 30] and algorithms to search process models that fit a particular profile [7, 10].

Against this background, business process model abstraction (BPMA) emerged as a technique reducing the number of models describing one business process at

different abstraction levels. BPMA is an operation on a business process model preserving essential process properties and leaving out insignificant process details in order to retain information relevant for a particular purpose. In practice there are various scenarios where BPMA is helpful [12, 15, 26]. A prominent BPMA use case is a construction of a process "quick view" for rapid process comprehension. In the "quick view" scenario the user wants to familiarize herself with a business process but has only a detailed model at hand. BPMA solves this problem by deriving a process model that specifies high-level activities and overall process ordering constraints.

The transition from the initial process model to the abstract one is realized by two primitive operations: elimination and aggregation [3, 22, 27]. While elimination simply drops model elements, aggregation merges a set of low-level model elements into one more general element. A special case of aggregation is activity aggregation that provisions coarse-grained activities for an abstract model. Often, there is a part-of relationship between the aggregated activities and the aggregating activity. While there is a number of papers elaborating on aggregation as the basic abstraction operation, all of them only consider the structure of the process model to decide which activities belong together. Therefore, also activities without semantic connection might be aggregated. However, to the best of our knowledge, there is no paper directly discussing how to aggregate activities which are close according to their domain semantics.

In this paper we develop an aggregation technique clustering activities according to their domain semantics. The technique can guide the user during a process model abstraction providing suggestions on related activities. To realize aggregation we refer to a part-of, or *meronymy*, relation between activities. The techniques we define can be extended towards composition, generalization, and classification [13, 20, 19]. The main contributions are the metric for comparing activity aggregations and the algorithm for aggregation mining. The presented approach is evaluated against a set of real-world process models.

The remainder of the paper is structured as follows. Section 2 illustrates the research problem. Section 3 presents an algorithm for derivation of activity aggregations. Section 4 describes the evaluation setting and results. In Section 5 we discuss the relation of our contribution to the existing research. Section 6 concludes the paper.

2 Motivation

A significant point of consideration in the design of the aggregation operation is the principle of activity aggregation. The principle defines which set of activities belong together and constitute a more coarse-grained activity. Existing BPMA techniques aggregate activities according to the process model structure: activities that belong to a process model fragment are aggregated into one coarse-grained activity. A fragment is either explicitly defined [3, 22], or identified by its properties [27]. To illustrate the shortcomings of purely structural BPMA techniques, let us refer to one concrete example, an approach developed in [27]. We demonstrate the abstraction mechanism by a process model example in Fig. 1. The model

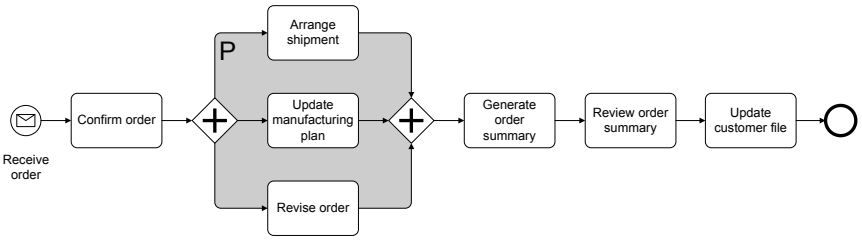

Fig. 1. "Customer order handling" business process

captures a business process of customer order handling at a factory. According to [27] the model is decomposed into several fragments: the shaded fragment P, and the top-level sequence that orders the start event, activity *Confirm order*, the fragment P, activities *Generate order summary*, *Review order summary*, *Update customer file*, and the end event. The abstraction algorithm enables aggregation of either a pair of sequential activities, or two branches in fragment P. Activity aggregation may continue and end up with a process model containing one coarse-grained activity. Obviously, this structural algorithm neglects the domain semantics of model elements. Consider activities *Generate order summary* and *Review order summary* in the example. The abstraction algorithm addresses none of the following questions:

1. Does an aggregation of *Generate order summary* and *Review order summary* makes sense from the point of view of the domain semantics?
2. Is aggregation of *Generate order summary* and *Review order summary* better than an aggregation of *Review order summary* and *Update customer profile*?

Apparently, structural information is insufficient to answer these questions. We consider domain semantics of activities as an alternative source of information. Intuitively, we may argue that it makes sense to aggregate *Generate order summary* and *Review order summary*, as both activities operate on object *order summary* and contribute to its evolution. We consider business process domain ontologies as a formal representation of the domain knowledge. In particular, we focus on ontologies that organize activities by means of a *meronymy* relation. Meronymy is a semantic relation denoting that one entity is a part of another entity. Meronymy is often referenced as a part-of relation. The meronymy relation organizes activities into a hierarchy, a *meronymy tree*. Activities at the top of the tree are more coarse-grained than those deeper in the tree. Given an activity in the meronymy tree, its direct descendents are low-level activities to be accomplished to fulfill the given activity. Hence, each non-leaf activity can be iteratively refined down to leaves. Consider an example meronymy tree in Fig. 2. According to the tree, to complete activity *Process order summary*, activities *Generate order summary* and *Review order summary* have to be executed.

We propose to use meronymy trees for activity aggregation. We reference a set of activities which is in question to be aggregated as an *aggregation candidate*. If all the activities of an aggregation candidate appear in a meronymy tree, they have a lowest common ancestor (LCA). We assume that the LCA can be used as a representative for the aggregation candidate. Returning to the example, we observe that *Generate order summary* and *Review order summary* are the direct descendents of activity *Process order summary* in the meronymy tree. According to our argument, this is a strong indication that these two activities should be aggregated. One can notice that *Update customer profile* appears in the tree as well. This allows us to consider the set *Generate order summary*, *Review order summary*, and *Update customer profile* as an aggregation candidate as well. Which of the two candidates is preferable? To answer this question we make an assumption that a good aggregation candidate comprehensively describes its LCA. In other words, we assume activities to be related, if they have a subsuming activity, LCA, and this activity is comprehensively described by the considered activities. According to this assumption, aggregation candidate *Generate order summary* and *Review order summary* is preferable, as it fully describes the ancestor *Process order summary*. At the same time the set *Generate order summary*, *Review order summary*, and *Update customer profile* does not provide a comprehensive description of *Create order report*: there are other activities in the meronymy tree that contribute to its execution. Following this argumentation we are able to mine activity aggregations in a process model and guide the user with recommendations on which activities belong together.

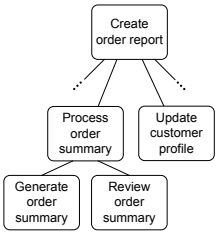

Fig. 2. Meronymy tree example

3 Meronymy-based Activity Aggregation Mining

This section formalizes the intuitive discussion sketched above. First, we introduce the basic concepts in subsection 3.1. Subsection 3.2 explains how activities in a business process model can be related to activities in a meronymy tree. Next, subsection 3.3 defines a metric enabling the comparison of aggregation candidates. Finally, subsection 3.4 defines an algorithm for activity aggregation mining.

3.1 Basic Concepts

We start by introducing the basic concepts. First, a universal alphabet of activities $\mathcal{A}$ is postulated. Further, we define the notion of a process model based on the general concepts of activities, gateways, and edges capturing the control flow. Notice that Definition 1 does not distinguish gateway types. We make this design decision, as we ignore the ordering constraints imposed by the control flow, but focus on the domain semantics of activities. Meanwhile, in subsection 3.4 we make use of distances between nodes, which motivates definition of a process model as a graph.

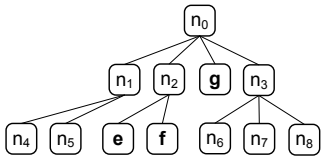

Fig. 3. Example meronymy tree t_1

Definition 1 (Process Model). A tuple $PM = (A, G, E)$ is a *process model*, where:

- $A \subseteq \mathcal{A}$ is the finite nonempty set of process model activities
- G is the finite set of gateways
- $A \cap G = \emptyset$ and $N = A \cup G$ is a finite set of nodes
- $E \subseteq N \times N$ is the flow relation, such that (N, E) is a connected graph.

An aggregation candidate $C \subseteq A$ is a subset of activities in a process model $PM = (A, G, E)$. The search for activity aggregations utilizes a domain ontology formalized as a meronymy forest.

Definition 2 (Meronymy Tree and Meronymy Forest). A *meronymy tree* is a tuple $t = (A_t, r_t, M_t)$, where:

- $A_t \subseteq \mathcal{A}$ is the finite non-empty set of activities
- $r_t \in A_t$ is the root
- $M_t \subseteq A_t \times (A_t \backslash \{r_t\})$ is the set of edges such that $(a, b) \in M_t$, if b is part of a
- M_t is an acyclic and coherent relation such that each activity $a \in A_t \backslash \{r_t\}$ has exactly one direct ancestor.

A meronymy forest F is a disjoint union of meronymy trees. We denote the set of activities in the meronymy forest as $A_F = \bigcup_{\forall t \in F} A_t$.

An example meronymy tree t_1 is presented in Fig. 3. Notice that according to the definition of a meronymy forest, each activity appears exactly in one meronymy tree. Definition 2 does not assume the existence of one super activity subsuming all the others. This is consistent, for instance, with ontologies, like the MIT Process Handbook, in which there are eight root activities [23].

Further we make extensive use of the lowest common ancestor notion. While this concept is well defined for a pair of nodes in the graph theory, we extend it to a node set. Let $t = (A_t, r_t, M_t)$ be a meronymy tree. We introduce an auxiliary function $lca_t : \mathcal{P}(A_t) \rightarrow A_t$, which for a set of activities $C \subseteq A_t$ returns a node $l \in A_t$ that is the lowest node having all the activities in C as descendants. The function is defined for a concrete meronymy tree and can be illustrated by the following two examples in tree t_1: $lca_{t_1}(\{e, f\}) = n_2$ and $lca_{t_1}(\{e, g\}) = n_0$.

3.2 Matching Activities: from Process Models to Meronymy Forest

To enable aggregation we need to relate process model activities to the information in an ontology, i.e., a meronymy forest. In the trivial case each process model

activity is captured in the ontology. However, in practice this is only the case if a process model is designed using the activities predefined by the ontology. However, we do not want to impose the restriction that a process model is constructed from activities in an ontology. Therefore, we use a matching step to determine which activity in a process model matches which activity in the meronymy forest. The matching step is particularly useful, if the process model and the meronymy forest have been designed independently, and now we would like to reuse the ontology for the activity aggregation problem.

Definition 3 (Activity Match and Activity Mix-Match). Let $PM = (A, G, E)$ be a process model, F be a meronymy forest. The function $match_{PM} : A \rightarrow \mathcal{P}(A_F)$ maps an activity to a set of activities in the meronymy forest. Function *match* is extended to sets such that $match_{PM} : \mathcal{P}(A) \rightarrow \mathcal{P}(\mathcal{P}(A_F))$ and for $Q \subseteq A$ it is defined as $match_{PM}(Q) = \{match_{PM}(q) |\ q \in Q\}$, which returns a set of match-sets, each corresponding to an element of Q.
Further, function $mixmatch_{PM}$ returns all potential combinations of matches for each process model activity from an input set. Function $mixmatch_{PM} : \mathcal{P}(A) \rightarrow \mathcal{P}(A_F)$ is defined so that for a set of activities $Q \subseteq A$ holds that $S \in mixmatch_{PM}(Q)$, if $|S| = |Q|$ and $\forall u, v \in S$ holds that $\exists_{a_1, a_2 \in A}[a_1 \neq a_2 \wedge u \in match_{PM}(a_1) \wedge v \in match_{PM}(a_2)]$.

The *match* mapping enables activity mapping in both cases: if the process model was designed in the presence of a meronymy forest, or independently. In the former case function *match* maps an activity to a trivial set, containing only this activity. In the latter case *match* maps a process model activity to a set of similar activities in the meronymy forest.

Various matching techniques exist [7, 8]. However, these techniques focus on matching activities from different process models, while we focus on matching activities from a process model to activities in an ontology. This means that we cannot exploit relations between activities, e.g., control flow, to establish a match. Therefore, we refine techniques for matching activities based only on their labels. To match an activity a in a process model to ontology activities, we consider activities with labels being most similar to the label of a. To address the similarity of labels we make use of relations between words, for instance, synonymy, antonymy, meronymy, and hyponymy, in the lexical database WordNet [25]. Given these relations, there are algorithms which enable finding a semantic similarity between words, e.g., see [4, 18, 21]. We use the similarity metric proposed by Jiang and Conrath in [18]. As the algorithms provide similarity values for words, we extend this method for labels as lists of words.

First, within each label all the words which are not verbs and nouns according to WordNet are removed. For each pair of labels all possible combinations of word to word mappings are considered. The mapping which results in a maximal similarity of words is considered. Each activity is mapped to the activities in the meronymy forest which are sufficiently similar to it. A configurable threshold value defines, when an activity is considered to be "sufficiently" similar to another one.

3.3 Aggregation Candidates Ranking

Without any prior knowledge, every subset of a process model activity set might be considered as a potential aggregation candidate. However, we aim to select only those aggregation candidates whose activities are strongly semantically related. There are various options for defining semantic relations among activities, for instance, based on operation on the same data object or execution by the same resource. In this paper we utilize meronymy relationships between activities in order to judge their semantic relatedness.

We say that activities in an aggregation candidate are strongly related, if together they comprehensively describe another activity—their LCA. The comprehensiveness depends on the existence of LCA descendants that do not belong to the aggregation candidate. The larger share of the LCA descendants belongs to the aggregation candidate, the more comprehensive is the description. For example, activity set $\{e, f\}$ in Fig. 3 fully describes its LCA n_2. In contrast, activity set $\{e, g\}$ describes only a small part of its LCA, activity n_0.

We define a metric to measure how comprehensively a set of activities describes its LCA. We impose the following requirements on the metric. The metric must reflect, whether the activities of an aggregation candidate describe the LCA comprehensively. The more descendants, which do not belong to the aggregation candidate, the LCA has, the smaller share of the LCA is described by the aggregation candidate. The metric must be neutral to the distance between activities of an aggregation candidate and the LCA, as the distance has no direct influence on how comprehensively activities describe their ancestors. Similarly, the relative position of the LCA to the tree root is not characteristic in this context. This position reflects the abstraction level of an activity. However, we have no interest in LCA abstraction level. We also require the metric to be neutral to the size of an aggregation candidate. Hence, the metric enables comparison of aggregation candidates of different sizes and even comparison of an aggregation candidate with aggregation candidates, which are its subsets. Finally, it is handy, if a metric has a value between 0 and 1. We summarize this requirements discussion as a list.

R1 Reflect, if the LCA has other descendents, except aggregation candidate activities.
R2 Be neutral to the depth of aggregation candidate in the LCA-rooted subtree.
R3 Be neutral to the depth of the LCA in the meronymy tree.
R4 Be neutral to the size of the aggregation candidate.
R5 Have a value between 0 and 1.

To present the designed function we introduce an auxiliary *meronymy leaves* first. The function sets up correspondence between a meronymy tree node and its descending leaves.

Definition 4 (Meronymy Leaves). Let $t = (A_t, r_t, M_t)$ be a meronymy tree in meronymy forest F. A function $w_t : A_t \to \mathcal{P}(A_t)$ is a *meronymy leaves function,* which for activity $a \in A_t$ returns the leaves of the subtree rooted to activity a.

Returning to the example tree t_1, consider $w_{t_1}(g) = \{g\}$ and $w_{t_1}(n_2) = \{e, f\}$. Given meronymy leaves function, we propose the following metric for aggregation candidate ordering.

Definition 5 (Degree of Aggregation Coverage). Let $t = (A_t, r_t, M_t)$ be a meronymy tree in a meronymy forest F and $C \subseteq A_t$ be an aggregation candidate. A function $cover : \mathcal{P}(A_t) \to (0, 1]$ describes the *degree of aggregation coverage,* defined as: $cover(C) = \dfrac{\left| \bigcup_{\forall a \in C} w_t(a) \right|}{|w_t(lca_t(C))|}$.

The metric captures the extent to which the activity set covers the LCA activity. The larger the share, the more "comprehensive description" provides the activity set. For the motivating example in the tree t_1 the metric has values $cover(\{e, f\}) = 1$ and $cover(\{e, g\}) = 0.25$, i.e., $cover(\{e, f\}) > cover(\{e, g\})$. Due to this we conclude that $\{e, f\}$ is a better aggregation than $\{e, g\}$. As the aggregation metric makes use of *meronymy leaves* function, it considers the presence of other LCA descendants rather than those in the aggregation candidate. As the metric makes no use of distance measures, it is neutral to depth of aggregation candidates in the LCA-rooted subtree, as well as the depth of the LCA in the meronymy tree. The metric is indifferent to the size of the aggregation candidate, but considers the number of leaves in the tree "covered" by the candidate. Finally, the metric value is always greater than 0, and reaches 1 at most. We conclude that the proposed aggregation metric satisfies requirements *R1–R5*.

3.4 Activity Aggregation Mining Algorithm

Finally, we propose an algorithm for mining of activity aggregations from a process model. The mining algorithm has two subproblems: generation of aggregation candidates out of a process model and selection of aggregations from aggregation candidates. While the latter problem exploits the developed aggregation metric *cover*, the former requires a discussion.

Generation of aggregation candidates from the model can be approached in a brute force fashion, if all the possible activity combinations are considered. However, the number of combinations in this case is $2^{|A|}$, where A is the set of activities in a process model. As BPMA addresses complex process models with a large number of activities, this brute force method is insufficient. We need a method for coping with the computational complexity. A wholesome observation is that related activities are co-located within a process model [29]. According to this observation, we assume that for a given activity, the related activities can be found within a fixed graph distance. In this way we effectively manage the computational complexity problem. The computational complexity is further reduced, if we iteratively construct aggregation candidates pruning redundant ones. First, all the aggregation candidates of size two are created and analyzed. Candidates, which matches do not appear in one meronymy tree, are pruned. In the next iteration aggregation candidates of size three are constructed from the

Algorithm 1 Activity aggregation mining

```
1: mine(Model PM = (A, G, E), MForest F, double cover_0, int dist)
2: Set aggregations = ∅;
3: for all activity ∈ A of PM do
4:    Set candidates = ∅;
5:    for all activityPair ∈ findNeighbours(activity, dist) do
6:       candidate = {activityPair[1], activityPair[2]};
7:       for all ontologyCandidate ∈ mixmatch_PM(candidate) do
8:          if ∃t ∈ F, t = (A_t, r_t, M_t) : ontologyCandidate ⊆ A_t then
9:             candidates = candidates ∪ {candidate};
10:            if cover(ontologyCandidate) ≥ cover_0 then
11:               aggregations = aggregations ∪ {candidate};
12:   aggregations = aggregations ∪ kStep(candidates, PM, F, cover_0, dist);
13: return aggregations;
14:
15: \\Inductive step of aggregation mining
16: kStep(Set kCandidates, Model PM, MForest F, double cover_0, int dist)
17: Set aggregations = ∅;
18: Set (k + 1)Candidates = ∅;
19: int k = kCandidates[1].size;
20: for all candidatePair from kCandidates do
21:    newCandidate = candidatePair[1] ∪ candidatePair[2];
22:    if newCandidate.size == k + 1 then
23:       for all ontologyCandidate ∈ mixmatch_PM(newCandidate) do
24:          if ∃t ∈ F, t = (A_t, r_t, M_t) : ontologyCandidate ⊆ A_t then
25:             (k + 1)Candidates = (k + 1)Candidates ∪ {newCandidate};
26:             if cover(ontologyCandidate) ≥ cover_0 then
27:                aggregations = aggregations ∪ {newCandidate};
28: aggregations = aggregations ∪ kStep((k+1)Candidates, PM, F, cover_0, dist);
29: return aggregations;
```

candidates of size two. Hence, the construction of aggregation candidates of size $k+1$ makes use of aggregation candidates of size k and their pruning.

Algorithm 1 formalizes the discussion above. The input of the algorithm is a process model $PM = (A, G, E)$, a meronymy forest F, an aggregation metric threshold value $cover_0$, and $dist$—the graph node distance. The threshold value $cover_0$ and distance $dist$ allow to set up the algorithm. The values can be selected by the user or empirically obtained (see Section 4). The output of the algorithm is the set of aggregations. The iterative construction of aggregations of increasing size is realized by two functions: *mine* and *kStep*. The entry point of the algorithm is function *mine*. For each activity in a process model (line 3) the algorithm finds a set of neighboring activities within a specified distance $dist$. Function $findNeigbours(activity, dist)$ returns the set of activities allocated within a distance not greater than $dist$ from $activity$ in the process model (line 5). Within this set all the subsets of size two are considered as aggregation candidates (line 6). Each candidate is evaluated against the ontology. If the candidate has

no mappings to the ontology activities that belong to one tree, it is pruned (lines 7–8). Otherwise, the candidate mappings are evaluated against the specified metric threshold value $cover_0$. If there is at least one mapping of an aggregation candidate, for which the value of *cover* is greater than m_0 the candidate is considered to be an aggregation (lines 10–11). All the aggregation candidates that have not been pruned are used as the input for function *kStep* (line 12). Function *kStep* iteratively increases the size of aggregation candidates by one, pruning and evaluating them (lines 16–29). The pruning and evaluation of candidates follows the same principle as in function *mine*.

4 Empirical Validation

We evaluated the developed aggregation technique by applying it to a set of models capturing the business processes of a large electronics manufacturer. The model collection considered in the experiment includes 6 business process models. Each model contains on average 42 activities, with a minimum of 18 activities and a maximum of 81 activities. On average, an activity label contains 4.1 words.

A meronymy forest is represented by the MIT Process Handbook [23]. The MIT Process Handbook describes business processes elicited by researchers in the interviews with business process experts. It spans several business domains, like sales, distribution, and production. The handbook describes about 5 000 activities and specifies hyponymy and meronymy relations between them. We make use of activities and a meronymy relation only. The process models were not aligned with the Handbook in advance: no relations between process model activities and the MIT Process Handbook activities were established. We matched process model activities to the activities of the handbook according to the semantics of their labels, as discussed in Section 3.2.

In the evaluation we rely on human judgment for assessing the relevance of the results delivered by our approach. We asked a process modeling expert from TU Eindhoven, who was unfamiliar with the technique, to select those aggregations that were considered relevant. We gave the instruction to consider an abstraction relevant, if a given set of activities could be reasonably "abstracted" into a single activity. The means for abstraction that could be considered were: aggregating the activities in the abstraction, generalizing the activities, putting the activities into a common subprocess, or any other means that the evaluator considered relevant.

Activity aggregations delivered by the aggregation mining algorithm vary in size. To get a precise result we decomposed each mined activity set into a set of all its subsets of size two. Due to this decomposition, the evaluation shows, if all the activities of the aggregation are strongly related. If in the set $\{a, b, c\}$ activity c is weakly related to a and b, the human easily points this out claiming (a, c) and (b, c) irrelevant.

We have conducted a series of experiments in which we varied the parameters of our aggregation technique. In each run of the experiments we have fixed the parameters of *match* function (each process model activity was mapped to at most 10 activities in the Handbook). At the same time we varied the node

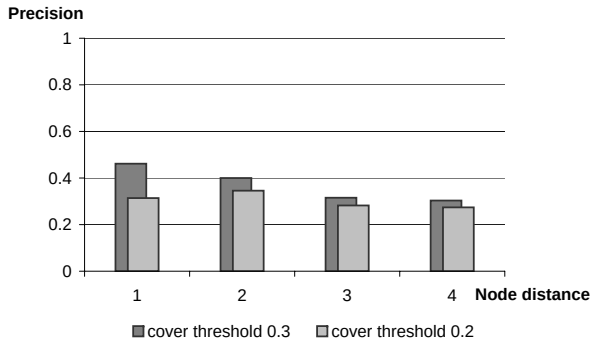

Fig. 4. Precision of the activity aggregation mining algorithm

distance and *cover* threshold value. The node distance runs the values from 1 to 4, while the *cover* threshold values were 0.2 and 0.3. Within the experiment we observed the precision value—a number of relevant activity pairs retrieved by the algorithm related to the total number of retrieved pairs.

Fig. 4 illustrates the observed results. The precision value varies between 0.27 (the node distance equals to 4 and the cover threshold value is of 0.2) and 0.46 (the node distance equals to 1 and the cover threshold value is of 0.3). One can see two tendencies in the experiment. First, a higher $cover_0$ threshold value leads to a higher precision. Indeed, a high threshold prunes more aggregation candidates, as it imposes more strict search conditions. The total number of aggregations declines, increasing the precision. Second, the increase of node distance leads to a precision decrease. This observation can be explained by the fact that a node distance increase brings more activities into algorithm's consideration. As [29] argues the greater the distance is, the less related activities appear in the set. Thereby, the precision decrease is expected.

While the technique returns a considerable amount of helpful suggestions, there is still quite a number of irrelevant aggregations proposed. Hence, we aim to improve the technique precision by combining it with other information on relatedness, e.g. shared data access or same resource execution. Further, the conducted experiment evaluated both aggregation mining algorithm and *match* function. Since the process models and the used ontology were not aligned beforehand, there is also a contribution to a gap in precision by the *match* technique. To study the behavior of aggregation mining algorithm further, we need a setting, where process models are created using activities from a domain ontology. We perceive such an evaluation as the future work. Concluding, we suggest that although the developed technique cannot be used in fully automatic fashion, it can support the user with suggestions.

5 Related Work

This paper extends existing methods for business process model abstraction by addressing the semantics of activities in business process models. As we mentioned

in Section 2, the available BPMA techniques focus on the structural aspects of process model transformation, e.g., see [3, 27, 31]. In addition to those papers, there are works that discuss criteria for abstraction of model elements, see [12, 15]. These criteria are defined on activity properties such as execution cost, duration, frequency, and associated resources; activities are either abstracted or preserved depending on whether they meet the given criteria. Basu and Blanning [2] propose a technique for detecting possible abstractions by searching for structural patterns. We are not aware of BPMA algorithms and, in particular, activity aggregation algorithms, that use semantic information. However, object-oriented analysis and design has extensively studied the meronymy, or whole-part, relation. In [1] Barbier et al. extend the formalization of this relation beyond the UML specification. Guizzardi focuses on the modal aspects of the meronymy relation and the underlying objects in [14].

Business process model abstraction is related to modularization of business process models, which is the task of extracting a set of activities and putting them into a common subprocess. In [28] Reijers and Mendling investigate quality metrics that determine whether tasks should be abstracted and put into a common subprocess. This work is extended by studying strict criteria that can be used to determine when tasks can be put into a common subprocess [29].

Another stream of the related research is the work on lexical semantic relatedness. There exist a number of measures that utilize semantic relations of WordNet to evaluate semantic relatedness of words. In [4] Budanitsky and Hirst provide a comprehensive overview of such measures that consider the WordNet hyponymy relation. Further, they evaluate the measures and conclude on their pros and cons. Our interest in this body of research is twofold. On the one hand, we apply these results for finding related activity labels: based on the outcome of evaluation in [4] and our experiment setting, we utilize the measures proposed in [18] and [21]. On the other hand, these measures inspired the aggregation metric that we developed in this paper. Other techniques for finding related activity labels or related process models vary with respect to the type of information that they rely on. Structural and semantic information is used in [11], behavioral information in [9], and graph-edit distance techniques are used as well [8]. However, these techniques focus on finding related activities from different processes, while the focus of our work is mining of related activities in the same model.

The discussion of semantic relations between activities falls into the area of semantic BPM. Hepp et al. [17] present a vision of semantic BPM, explaining the role of ontologies and showing how they expand process analysis capabilities. In [5] Casati et al. suggest to use taxonomies for high-level analysis of business processes. Medeiros et al. employ activity meronymy and hyponymy relations to advance analysis of information derived from logs [24]. Although the above named papers describe advanced analysis of process models, they do not suggest metrics that enable mining of aggregations in domain ontologies. Finally, Dapoigny and Barlatier introduce a formalism that facilitates a precise and unambiguous description of a part-whole relation [6]. Similar to the semantic BPM work, their contribution does not directly address the aggregation mining problem.

6 Conclusions and Future Work

This paper presented a semi-automated approach to aggregation of activities in process models. We developed a technique for mining sets of semantically related activities in a process model. The activity relatedness is evaluated with the help of information external to the model—a domain ontology specifying activity meronymy relation. The paper contributions are 1) the metric enabling judgment on relatedness of activity sets and 2) the algorithm for activity aggregation mining. Further, we proposed a technique for matching activities of a process model and an ontology. The developed approach is evaluated against a real-world setting including a set of business process models from a large electronics manufacturer and the domain ontology of the MIT Process Handbook.

We foresee a number of future research directions. First, there is potential for further improvement of the activity matching technique. In particular, we aim to develop advanced label analysis techniques by identifying verbs and business objects. Second, we want to evaluate the aggregation mining technique in a setting where process models have been created using a domain ontology. Such an evaluation allows to check the performance of the aggregation mining algorithm in isolation. Third, it is of great interest to investigate other types of information in process models that may support aggregation, for instance, data flow or organizational role hierarchies. Finally, our research agenda includes a study of interconnection between structural and semantic aggregation approaches.

References

1. F. Barbier, B. Henderson-Sellers, A. Le Parc-Lacayrelle, and J.-M. Bruel. Formalization of the Whole-Part Relationship in the Unified Modeling Language. *IEEE TSE*, 29(5):459–470, 2003.
2. A. Basu and R.W. Blanning. Synthesis and Decomposition of Processes in Organizations. *ISR*, 14(4):337–355, 2003.
3. R. Bobrik, M. Reichert, and T. Bauer. View-Based Process Visualization. In *BPM 2007*, pages 88–95, Brisbane, Australia, 2007. Springer.
4. A. Budanitsky and G. Hirst. Evaluating WordNet-based Measures of Lexical Semantic Relatedness. *COLI*, 32(1):13–47, 2006.
5. F. Casati and M.-Ch. Shan. Semantic Analysis of Business Process Executions. In *EDBT 2002*, pages 287–296, London, UK, 2002. Springer.
6. R. Dapoigny and P. Barlatier. Towards an Ontological Modeling with Dependent Types: Application to Part-Whole Relations. In *ER 2009*, pages 145–158, Gramado, Brazil, 2009. Springer.
7. R. M. Dijkman, M. Dumas, and L. García-Bañuelos. Graph Matching Algorithms for Business Process Model Similarity Search. In *BPM 2009*, pages 48–63, Ulm, Germany, 2009. Springer.
8. R. M. Dijkman, M. Dumas, L. García-Bañuelos, and R. Käärik. Aligning Business Process Models. In *EDOC 2009*, pages 45–53, 2009.
9. B. van Dongen, R. M. Dijkman, and J. Mendling. Measuring Similarity between Business Process Models. In *CAiSE 2008*, pages 450–464. Springer, 2008.
10. M. Dumas, L. García-Bañuelos, and R. M. Dijkman. Similarity Search of Business Process Models. *IEEE Data Eng. Bull.*, 32(3):23–28, 2009.

11. M. Ehrig, A. Koschmider, and A. Oberweis. Measuring Similarity between Semantic Business Process Models. In *APCCM 2007*, pages 71–80, Ballarat, Victoria, Australia, 2007. ACSC.
12. R. Eshuis and P. Grefen. Constructing Customized Process Views. *DKE*, 64(2):419–438, 2008.
13. J. Evermann and Y. Wand. Toward Formalizing Domain Modeling Semantics in Language Syntax. *IEEE TSE*, 31(1):21–37, 2005.
14. G. Guizzardi. Modal Aspects of Object Types and Part-Whole Relations and the *de re/de dicto* Distinction. In *CAiSE*, volume 4495 of *LNCS*, pages 5–20. Springer, 2007.
15. C. W. Günther and W. M. P. van der Aalst. Fuzzy Mining—Adaptive Process Simplification Based on Multi-perspective Metrics. In *BPM 2007*, pages 328–343, Berlin, 2007. Springer.
16. A. Hallerbach, T. Bauer, and M. Reichert. Capturing Variability in Business Process Models: The Provop Approach. *SPIP*, 2009.
17. M. Hepp, F. Leymann, J. Domingue, A. Wahler, and D. Fensel. Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. In *ICEBE 2005*, pages 535–540. IEEE CS, 2005.
18. J. J. Jiang and D. W. Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *ROCLING 1997*, pages 19–33, 1997.
19. T. Kühne. Matters of (Meta-) Modeling. *SoSyM*, 5(4):369–385, 2006.
20. T. Kühne. Contrasting Classification with Generalisation. In *APCCM 2009*, Wellington, New Zealand, January 2009.
21. D. Lin. An Information-Theoretic Definition of Similarity. In *ICML 1998*, pages 296–304. Morgan Kaufmann, 1998.
22. D. Liu and M. Shen. Workflow Modeling for Virtual Processes: an Order-preserving Process-view Approach. *IS*, 28(6):505–532, 2003.
23. Th. W. Malone, K. Crowston, and G. A. Herman. *Organizing Business Knowledge: The MIT Process Handbook*. The MIT Press, Cambridge, MA, USA, 2003.
24. A. K. A. De Medeiros, W. M. P. van der Aalst, and C. Pedrinaci. Semantic Process Mining Tools: Core Building Blocks. In *ECIS 2008*, pages 1953–1964, Galway, Ireland, 2008.
25. A. G. Miller. Wordnet: A Lexical Database for English. *CACM*, 38(11):39–41, 1995.
26. A. Polyvyanyy, S. Smirnov, and M. Weske. Process Model Abstraction: A Slider Approach. In *EDOC 2008*, pages 325–331. IEEE CS, 2008.
27. A. Polyvyanyy, S. Smirnov, and M. Weske. The Triconnected Abstraction of Process Models. In *BPM 2009*, pages 229–244, Ulm, Germany, 2009. Springer.
28. H. A. Reijers and J. Mendling. Modularity in Process Models: Review and Effects. In *BPM 2008*, pages 20–35, Milan, Italy, 2008. Springer.
29. H. A. Reijers, J. Mendling, and R. M. Dijkman. On the Usefulness of Subprocesses in Business Process Models. BPM Center Report BPM-10-03, BPMcenter.org, 2010.
30. M. Rosemann and W. M. P. van der Aalst. A Configurable Reference Modelling Language. *IS*, 32(1):1–23, 2007.
31. S. Smirnov. Structural Aspects of Business Process Diagram Abstraction. In *BPMN 2009*, pages 375–382, Vienna, Austria, 2009.