From Fine-Grained to Abstract Process Models: A Semantic Approach

Sergey Smirnov[a,*], Hajo A. Reijers[b], Mathias Weske[a]

[a] *Hasso Plattner Institute, Prof.-Dr.-Helmert-Str. 2-3, D-14482 Potsdam, Germany*
[b] *Eindhoven University of Technology, School of Industrial Engineering, PO Box 513, NL-5600 MB Eindhoven, The Netherlands*

Abstract

Business processes models easily become large and difficult to understand. Business process model abstraction has proven to be effective to generate readable, high-level views on business process models showing coarse-grained activities and leaving out irrelevant details. Yet, it is an open question how to perform abstraction in the same skillful way as experienced modelers combine activities into more abstract tasks. This article presents an approach that uses *semantic* information of a process model to decide on which activities belong together. The approach extends beyond the existing work that merely exploits the structural characteristics of a process model. The contribution of this article is twofold: we design a novel activity aggregation method and suggest how to discover the activity aggregation habits of human modelers. In an experimental validation, we use an industrial process model repository to compare the developed activity aggregation method with actual modeling decisions, and observe a strong correlation between the two. The presented work is expected to contribute to the development of modeling support for the application of effective process model abstraction, in this way improving the usability of business process models in practice.

Keywords: business process modeling, process model management, business process model abstraction

1. Introduction

Business process models are used within a range of organizational initiatives [20]. However, human readers are limited in their cognitive capabilities to make sense of large and complex business process models [2, 37]. One well-known way to address this issue is by applying *business process model abstraction*. This

*Corresponding author
Email addresses: sergey.smirnov@hpi.uni-potsdam.de (Sergey Smirnov), h.a.reijers@tue.nl (Hajo A. Reijers), mathias.weske@hpi.uni-potsdam.de (Mathias Weske)

is the act of retaining essential properties of a process model on a particular level of analysis, while simultaneously hiding insignificant process details from that level. Indeed, in a recent empirical investigation into the need for business process model abstraction [36], we found that its most prominent use case is the need to gain a so-called *quick overview* of the process. In such a situation, the user wants to familiarize herself in a short amount of time with a business process although what she has at her disposal is a large process model that consists of dozens of fine-grained activities. To address this use case, the process model can be displayed as a partially ordered set of coarse-grained activities, each of which aggregates a number of lower-level activities. As an example, an abstraction of a relatively fine-grained process model, which captures the creation of a forecasting report, is shown in Figure 1. This figure presents three process models formalizing the process at different levels of abstraction: model m is detailed, m_a is more abstract, while m'_a is the most abstract. We use the color coding to visualize how coarse-grained activities of model m_a are refined by activities in model m. For instance, activity *Perform quick analysis* is refined by activities *Prepare data for quick analysis* and *Perform quick data analysis*.

While it has been empirically established that abstraction can significantly improve the sense-making of large process models [29], a limited insight exists into the criteria that experienced modelers use to decide on which activities to aggregate into new ones. A number of techniques has been proposed that exploit *structural* properties of a process model to arrive at a more abstract one [7, 28]. Yet, it seems likely that experienced process modelers take a wider range of properties into account than just a model's control flow. For example, the fact that two activities use the same document and are executed by the same role may suggest that these can be conveniently clustered into one aggregated activity. This situation applies, for example, to the activities *Prepare data for quick analysis* and *Perform quick data analysis* of model m in Figure 1.

In this paper, we complement the existing streams of work with respect to process model abstraction by proposing an abstraction technique that incorporates semantic aspects contained within a process model. We rely on the observation that industrial process models are often enriched with elements that transcend control-flow information. Examples are: data that is being processed within an activity, IT systems invoked within particular activities, and roles assigned to activities. The central idea in this paper is that activities that are associated with the *same* non-control flow elements are semantically *related* and, therefore, are more appropriate candidates to be aggregated into the same activity than activities that do no share such elements.

A number of contributions has recently emerged that considers *semantic* aspects for aggregation, e.g., [9, 35]. However, their assumptions, e.g., the existence of an activity ontology [35], seem too strict for generic use. Our approach is based on the reuse of the frequently employed *vector space model*. This is an algebraic model, which is very popular in information retrieval [32]. As will be discussed in this paper, the use of vector spaces allows for determining the degree of similarity between activities according to several information types available in process models. We have validated the proposed technique by

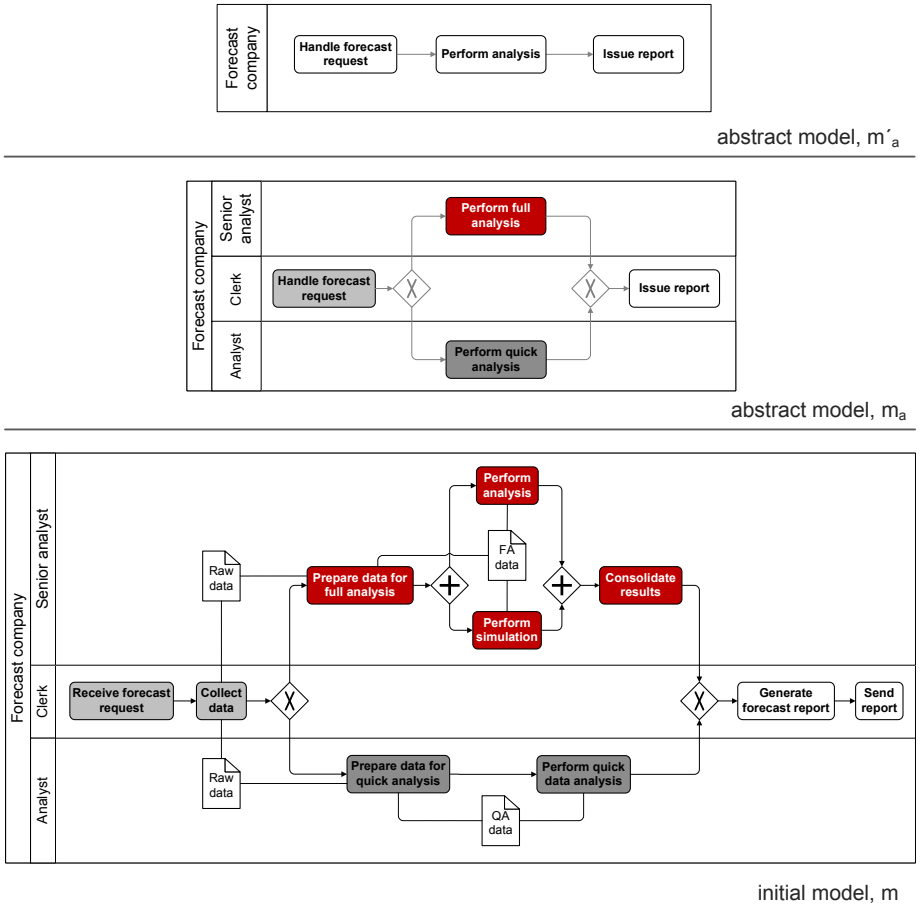


Figure 1: Motivating example: initial model and its abstract counterpart

applying it to a process model repository that is in use by a large European telecommunication organization. On the one hand, the repository incorporates hierarchical relations between high-level activities and the activities that they aggregate, which provides a view on how aggregation is given shape in a real situation. On the other hand, the contained process models contain various types of semantic information, which allows us to experiment with exploiting similarities between activities to come up with decisions on how to cluster these into more abstract tasks. The performed validation suggests that the developed approach closely approximates the decisions of the involved modelers to cluster activities according to their semantic similarities.

The contribution of this paper is twofold. First, we develop a novel method for activity aggregation analyzing activity environment within a process model. This method discovers sets of related activity, where each set corresponds to one coarse-grained activity of an abstract process model. Second, the arti-

cle suggests a technique that mimics the abstraction decisions of experienced modelers as discovered from sets of related, existing models. The technique allows to reuse the principles behind activity aggregation decisions for future aggregation decisions. As such, it can support novice process modelers that wish to make abstractions of complex, fine-grained process models. Since the lack of experienced process modelers is a noted issue in many large modeling projects [30], this is a valuable asset to improve the usability of process models in real-life applications. Furthermore, the proposed technique can also be of value to more experienced modelers when it is applied to enable those process model abstraction decisions that comply with their specific abstraction style. In this way, experts can accelerate their modeling routine while staying in control over the modeling outcome. Finally, the technique can also be used to safeguard a particular "fingerprint" of a process model collection by enforcing mutually consistent abstraction decisions.

The article is structured as follows. In Section 2, we provide the foundational concepts for our approach. Section 3 explains how activity aggregation can be interpreted as a cluster analysis problem. In Section 4 we discuss how the style of a business process designer making abstraction decisions can be discovered and captured. Section 5 empirically validates the proposed approach by using the industrial set of process models. Finally, Section 6 contrasts our contribution with the related research, while Section 7 concludes the article.

2. Foundations

This section introduces the key concepts of the developed activity clustering approach. We begin formally defining an environment of activities within process models, i.e., non-control flow elements. Further, we provide our notion of a process model establishing a link between activities and their environment. Finally, we postulate the idea of a process model collection with a subprocess refinement.

The designed aggregation algorithm inspects an activity environment, i.e., process model elements that are related to activities in a process model. Examples of such elements are *data objects* accessed by activities and *roles* supporting activity execution, e.g., see model in Figure 1. The list of such model element types varies depending on the process modeling language, the tool at hand, modeling procedures taken into account, and the modeler's style. Each of the model element types may be considered as an *activity property* that has a specific value. Definition 1 formalizes this concept.

Definition 1 (Activity Property Value and Activity Property Type). Let $\mathcal{P}$ be a finite nonempty set of activity property values. Alongside, $\mathcal{T}$ is a finite nonempty set of activity property types. The mapping $type : \mathcal{P} \rightarrow \mathcal{T}$ assigns a type to each value.

The process model in Figure 1 illustrates Definition 1. *Raw data*, *FA data* and *Analyst* are examples of activity property values. The process model presents

two activity property types: *Role* and *Data object*. For instance, *type(Raw data) = Data object*, *type(FA data) = Data object*, and *type(Analyst) = Role*. Further, we define a process model as follows.

Definition 2 (Process Model). A tuple $m_i = (A_i, G_i, F_i, P_i, props_i)$ is a *process model,* where:

- A_i is a finite nonempty set of activities;
- G_i is a finite set of gateways;
- $N_i = A_i \dot{\cup} G_i$ is the set of nodes, where $\dot{\cup}$ denotes a disjoint union of sets;
- $F_i \subseteq N_i \times N_i$ is the flow relation;
- $P_i \subseteq \mathcal{P}$ is a set of activity property values;
- $props_i : A_i \to 2^{P_i}$ is a mapping that assigns property values to an activity.

Definition 2 does not make a distinction between different gateway types since the future discussion does not make use of such a distinction. Mapping *props* assigns activity property values to model activities. Referring to model m in the motivating example of Figure 1, mapping *props* can be illustrated as *props(Collect data)* = {*Clerk, Raw data*}. Notice that Definitions 1 and 2 allow to manage the considered activity property types in a flexible fashion: It is sufficient to introduce a new activity property type to set $\mathcal{T}$, the values to $\mathcal{P}$, and respectively update mapping *type*. Thereafter, new activity properties can be easily considered within the activity aggregation. Finally, we postulate the concept of a process model collection.

Definition 3 (Process Model Collection). A tuple $c = (M, A, P, \sigma)$ is a *process model collection,* where:

- M is a nonempty finite set of n process models with elements $m_i = (A_i, G_i, F_i, P_i, props_i)$, where $i = 1, 2, \ldots, n$;
- $A = \dot{\cup}_{i=1,2,\ldots,n} A_i$ is a set of activity collections;
- $P = \cup_{i=1,2,\ldots,n} P_i$ is a set of sets of activity property values;
- $\sigma \subseteq M \times M$ is a subprocess relation refining a process model with subprocess models, such that $\forall m_i, m_j \in M$, where $j = 1, 2, \ldots, n$ and $i \neq j$, if $(m_i, m_j) \in \sigma$ then $(m_j, m_i) \notin \sigma^+$, where σ^+ is a transitive reflexive closure of σ.

Definition 3 explicitly enumerates the model collection activities and property value types. The relation σ formalizes the subprocess relation that exists between models. Note that according to the definition, σ enables only a process model hierarchy without loops. Without loss of generality we discuss abstraction of process models in the remainder of this paper within a process model collection. Indeed, a single process model m_i can be seen as a trivial process model collection $c = (\{m_i\}, A_i, P_i, \emptyset)$.

3. Activity Aggregation

This section presents the first core contribution of the article—an activity aggregation method based on cluster analysis. We start explaining how activity

aggregation can be interpreted as a cluster analysis problem in Section 3.1. Section 3.2 advocates several methods for the definition of activity distance measures.

3.1. Activity Aggregation as Cluster Analysis Problem

In this paper we interpret activity aggregation as a problem of cluster analysis. Consider process model $m_i = (A_i, G_i, F_i, P_i, props_i)$ from process model collection $c = (M, A, P, \sigma)$. The set of objects to be clustered is the set of activities A_i. The objects are clustered according to a distance measure: Objects that are "close" to each other according to this measure are put together. The distance between objects is evaluated through an analysis of activity property values P. The cluster analysis' outcome, activity clusters, correspond to coarse-grained activities within the abstract process model. While cluster analysis provides a large variety of algorithms, e.g., see [33], we focus on one algorithm that suits the business process model abstraction use case in focus.

In the considered scenario, the user demands control over the number of activities in the abstract process model. For example, a popular practical guideline is that five to seven activities are displayed on each level in the process model [34]. Provided such a fixed number, e.g., 6, the clustering algorithm has to ensure that the number of clusters equals the request by the user. We turn to the use of the k-means clustering algorithm: It is simple to implement and typically exhibits good performance [17]. K-means clustering partitions an activity set into k clusters. The algorithm assigns an activity to that cluster of which the centroid is the closest to this activity. To evaluate an activity distance, we analyze activity property values P. Since a number of alternative activity distance measures can be foreseen, we elaborate on these in the next section.

3.2. Activity Distance Measures

To introduce the distance measure among activities we represent activities as vectors in a vector space. Such an approach is inspired by algebraic models that are widely used in information retrieval [3]. Section 3.2.1 discusses the adoption of a simple algebraic model, the vector space model (VSM) [32]. While the VSM facilitates an activity distance evaluation, it does not allow to capture semantic relations between activity property values. The advanced vector space models described in Section 3.2.2 overcome this limitation. Finally, Section 3.2.3 provides an alternative classification of vector space models based on types of activity property values.

3.2.1. Vector Space Model

In information retrieval, the VSM allows to compare documents with respect to the words they contain. Two documents containing significantly overlapping word sets are highly similar, while documents with a small intersecting word set have a low similarity. The VSM assumes no linguistic relation between words, e.g., no synonymy, meronymy, or hyponymy is considered. This assumption allows to formalize distinct words as the dimensions of the vector space. To adapt the VSM

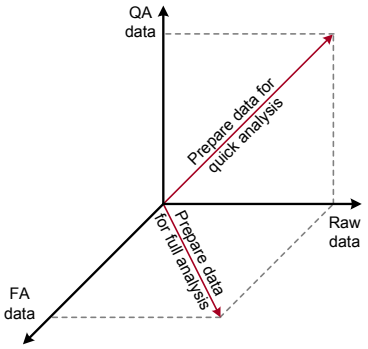


Figure 2: Example of a vector space formed by dimensions *FA data, QA data, Raw data*

to activity clustering, we assume that any two distinct activity property values are independent: There is no semantic relation between them. For instance, the roles *analyst* and *senior analyst* have nothing in common. Due to this, the space dimensions correspond to activity property values P and the vector space can be captured as a vector $(p_1, \ldots, p_{|P|})$, where $p_j \in P$ for $j = 1, \ldots, |P|$. Consider an example set of property values $P' = \{$*FA data, QA data, Raw data*$\}$ and the corresponding vector space presented in Figure 2. A vector $\vec{v_a}$ representing an activity $a \in A_i$ in process model $m_i = (A_i, G_i, F_i, P_i, props_i)$ is constructed as follows. If activity a is associated with a property value $p_j \in P_i$, the corresponding vector dimension $\pi_j(\vec{v_a})$ has value 1; otherwise, the dimension $\pi_j(\vec{v_a})$ has value 0:

$$\pi_j(\vec{v_a}) = \begin{cases} 1, & \text{if } p_j \in props_i(a); \\ 0, & \text{otherwise.} \end{cases}$$

For the process model m in Figure 1, activities *Prepare data for quick analysis* and *Prepare data for full analysis* respectively correspond to vectors $\vec{v_1} = (0, 1, 1)$ and $\vec{v_2} = (1, 0, 1)$ in the vector space with dimensions *FA data, QA data, Raw data*, see Figure 2.

The similarity of two vectors in the space is defined by the angle between these vectors: The larger the angle, the more distant the activities are. Typically, the cosine of the angle between two vectors is used as a vector similarity measure:

$$sim(a_1, a_2) = cos(\vec{v_{a_1}}, \vec{v_{a_2}}) = \frac{\vec{v_{a_1}} \cdot \vec{v_{a_2}}}{\|\vec{v_{a_1}}\| \, \|\vec{v_{a_2}}\|}.$$

Then, the distance between the two activities is:

$$dist(a_1, a_2) = 1 - sim(a_1, a_2).$$

By construction, the vector dimension values are non-negative. Hence, the activity similarity and activity distance measures vary within the interval $[0, 1]$. Returning to the example with activities *Prepare data for full analysis* (a_1) and *Prepare data for quick analysis* (a_2) we observe the following similarity value:

$$sim(a_1, a_2) = \frac{\vec{v_{a_1}} \cdot \vec{v_{a_2}}}{||\vec{v_{a_1}}|| \cdot ||\vec{v_{a_1}}||} = \frac{1}{\sqrt{2} \cdot \sqrt{2}} = \frac{1}{2}.$$

In this case, the distance has the according value:

$$dist(a_1, a_2) = 1 - sim(a_1, a_2) = 1 - \frac{1}{2} = \frac{1}{2}.$$

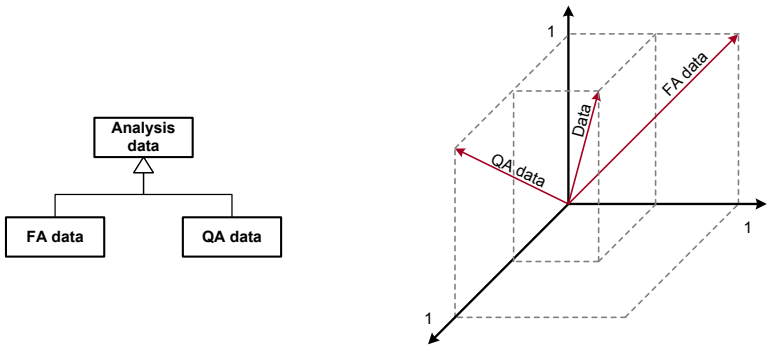
3.2.2. Advanced Vector Space Models

The previous section argued how the VSM, the simple algebraic model, realizes an activity distance measure. Recall that the VSM assumes the independence of activity property values. To reflect this fact the VSM models activity property values as the dimensions of the space, i.e., activity property values are modeled by orthogonal vectors. However, in practice activity property values are often related. For instance, data objects of the business process can be bound by *is-a* and *part-of* relations. UML 2.0 class diagrams allow to express such relations [27]. Figure 3(a) provides an example of the UML class diagram organizing *Analysis data*, *FA data*, and *QA data* by a *generalization* (*is-a*) relation. In the diagram *Analysis data* is the most generic concept refined by two concepts: *FA data* and *QA data*. In the described setting, *FA data* and *QA data* are related by a common parent *Analysis data*. Obviously, the VSM does not reflect this relation.

To overcome the limitation of the VSM we consider advanced algebraic models, like the *generalized vector space model* (GVSM) [40], the *topic vector space model* (TVSM) [5], and the *enhanced topic vector space model* (eTVSM) [24]. These models build on the basic idea behind the VSM, but reflect the relations between activity property values. We choose to use the eTVSM to model the semantic relations between activity property values. The advantage of the eTVSM is that it addresses activity property value relations and specifies how to capture them in a vector space. The key concept of the eTVSM is a *topic*. In our setting, topics correspond to activity property values. For instance, data objects *FA data*, *QA data*, and *Analysis data* are topics. The topics are organized into a hierarchy by a relation. The eTVSM names such a hierarchy a *topic map*. Figure 3(a) shows the class diagram containing three data objects and the generalization relation between them. This class diagram can be read as a topic map, since it organizes elements into a hierarchy. We formalize a relation between the topics in a topic map by a *super topic function* $S : P \to 2^P$. S maps an activity property value to its direct parent in the topic map. Note that the definition of a super topic function S allows for multiple parents of one topic. As an example, we observe in Figure 3(a):

- $S(FA\ data) = \{Analysis\ data\}$,
- $S(QA\ data) = \{Analysis\ data\}$.

Following [24] we extend the mapping S to a transitive super topic mapping for the k-th level $S^k : P \to 2^P$ that maps an activity property value to the set containing its parents at the k higher level:



(a) The UML 2.0 class diagram organizing three types of data by generalization relation

(b) Representation of the three activity property values in the eTVSM

Figure 3: A topic map formalized as the UML 2.0 class diagram and the corresponding representation of the topics in the eTVSM

$$S^k(p_i) = \begin{cases} S(p_i), & \text{if } k = 1; \\ \bigcup_{p_j \in S^{k-1}(p_i)} S(p_i), & \text{if } k > 1. \end{cases}$$

We also postulate an unbounded transitive super topic mapping $S^* : P \to 2^P$ so that:

$$S^*(p_i) = \bigcup_{j=1}^{n} S^j(p_i)$$

where n is the height of the topic map. We require the mappings S, S^k, and S^* to be irreflexive. The mapping S distinguishes the set of leaves among activity property values $P_L = \{p_i \in P | \nexists p_k \in P, p_i \in S(p_j)\}$. In Figure 3(a) *FA data* and *QA data* are leaves.

The eTVSM represents topics in the vector space as *topic vectors* and prescribes a construction method for them. A topic vector $p_i \in P_L$ is defined as follows:

$$\vec{p_i} = [(p^*_{i,1}, p^*_{i,2}, \ldots, p^*_{i,|P|})]_1, \text{ where}$$

$$p^*_{i,d} = \begin{cases} 1, & \text{if } p_d \in S^*(p_i) \vee i = d; \\ 0, & \text{else.} \end{cases}$$

According to this definition a vector has the dimensionality of the topic set size, $|P|$, since each topic corresponds to a dimension. A vector that models topic p_i has a length of 1 along a dimension if one of the two conditions holds. First, this dimension corresponds to the topic p_j that is the super topic of p_i. Second, the dimension corresponds to the topic p_i. In any other case, the vector has the

	Analysis data	FA data	QA data
Analysis data	1.00	0.87	0.87
FA data	0.87	1.00	0.50
QA data	0.87	0.50	1.00

Table 1: Similarities between topic vectors corresponding to *Analysis data*, *FA data*, and *QA data* derived according to the topic map in Figure 3(a)

length of 0 along the dimension. For the example in Figure 3(a) the activity property values are modeled as the following vectors:

$$\begin{aligned}\vec{p_1} &= [(1,0,0)]_1 = (1,0,0)\\ \vec{p_2} &= [(1,1,0)]_1 = (\tfrac{1}{\sqrt{2}},\tfrac{1}{\sqrt{2}},0)\\ \vec{p_3} &= [(1,0,1)]_1 = (\tfrac{1}{\sqrt{2}},0,\tfrac{1}{\sqrt{2}})\end{aligned}$$

Table 1 presents the similarity measure of the three activity property values *Analysis data*, *FA data*, and *QA data* found using the eTVSM. In contrast to the VSM, the eTVSM allows activity property values with a similarity different from zero. Finally, a vector modeling an activity in the space is defined as the sum of topic vectors. The sum is over those topic vectors that correspond to activity property values associated with the activity a:

$$\vec{v_a} = \sum_{p \in props_i(a)} \vec{p}.$$

To complete the illustration of the eTVSM we return to the example with activities *Prepare data for quick analysis* and *Prepare data for full analysis*. We evaluate the similarity of two activities given the process model m in Figure 1 and the topic map in Figure 4. First, we derive topic vectors:

$$\begin{aligned}\vec{p_1} &= [(1,0,0,0,0)]_1 = (1,0,0,0,0)\\ \vec{p_2} &= [(1,1,0,0,0)]_1 = (\tfrac{1}{\sqrt{2}},\tfrac{1}{\sqrt{2}},0,0,0)\\ \vec{p_3} &= [(1,0,1,0,0)]_1 = (\tfrac{1}{\sqrt{2}},0,\tfrac{1}{\sqrt{2}},0,0)\\ \vec{p_4} &= [(1,0,1,1,0)]_1 = (\tfrac{1}{\sqrt{3}},0,\tfrac{1}{\sqrt{3}},\tfrac{1}{\sqrt{3}},0)\\ \vec{p_5} &= [(1,0,1,0,1)]_1 = (\tfrac{1}{\sqrt{3}},0,\tfrac{1}{\sqrt{3}},0,\tfrac{1}{\sqrt{3}})\end{aligned}$$

The vectors $\vec{a_1}$ and $\vec{a_2}$ capturing, respectively, activities *Prepare data for quick analysis* and *Prepare data for full analysis* are found as follows:

$$\begin{aligned}\vec{a_1} &= \vec{p_2} + \vec{p_4} = [(2,1,1,1,0)]_1 = (\tfrac{2}{\sqrt{7}},\tfrac{1}{\sqrt{7}},\tfrac{1}{\sqrt{7}},\tfrac{1}{\sqrt{7}},0)\\ \vec{a_2} &= \vec{p_2} + \vec{p_5} = [(2,1,1,0,1)]_1 = (\tfrac{2}{\sqrt{7}},\tfrac{1}{\sqrt{7}},\tfrac{1}{\sqrt{7}},0,\tfrac{1}{\sqrt{7}})\end{aligned}$$

Having the two vectors modeling activities we use them to evaluate the activity similarity according to the eTVSM:

$$sim(a_1,a_2) = cos\alpha = \frac{\vec{v_{a_1}} \cdot \vec{v_{a_2}}}{||\vec{v_{a_1}}|| \cdot ||\vec{v_{a_1}}||} = \frac{6}{\sqrt{7}\cdot\sqrt{7}} = \frac{6}{7}.$$

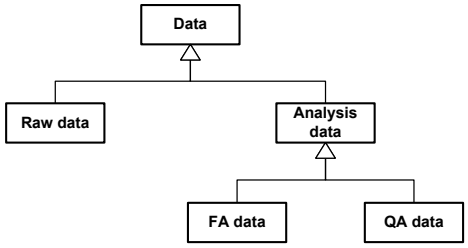


Figure 4: A class diagram capturing is-a relation between data objects of the process model

Note that the similarity measure for the activities *Prepare data for quick analysis* and *Prepare data for full analysis* varies in two vector spaces: While in the VSM the similarity is $\frac{1}{2}$, in the eTVSM the similarity measure increases to $\frac{6}{7}$. In this way the semantic relations existing between the data objects have an impact on the similarity measure. For instance, the closer proximity of activities *Prepare data for quick analysis* and *Prepare data for full analysis* allows to achieve aggregation shown in model m'_a in Figure 1. Both activities are aggregated to one coarse-grained activity *Perform analysis*.

3.2.3. Vector Space Model Types

For a process model collection $c = (M, A, P, \sigma)$ we distinguish two types of vector spaces. On the one hand, a vector space can be formed by the dimensions corresponding to the activity property values while disregarding their type, i.e., all elements of P. We refer to such spaces as *heterogeneous vector spaces*. An example of a heterogeneous vector space is a space with 6 dimensions *Analyst, Clerk, FA data, QA data, Raw data*, and *Senior analyst*. On the other hand, a vector space can be formed by the dimensions corresponding to the activity property values of a particular type. Given an activity property type t, such a space is formally defined by the set $P_t = \{\forall p \in P : type(p) = t\}$. We refer to such spaces as *homogeneous vector spaces*. Figure 2 provides an example of a homogeneous vector space formed by activity properties of type *Data object*. We denote the activity distance in a homogeneous space with $dist_h(a_1, a_2)$ and in a heterogeneous vector space with $dist_t(a_1, a_2)$, where t is the respective activity property type. Both distance measures can be employed for activity aggregation. If the user wants to make use of one activity property type t only, the distance is defined by $dist_t$. To cluster activities according to several activity property types, $dist_h$ can be employed. In addition, we introduce an alternative distance measure $dist_{agg}$ that aggregates multiple homogeneous distance measures $dist_t$:

$$dist_{agg}(a_1, a_2) = 1 - \frac{1}{|T|} \sum_{\forall t \in T} w_t \cdot dist_t(a_1, a_2) \quad (1)$$

In Equation 1, the set T corresponds to the activity property types that appear in process model collection c. Then, function $dist_{agg}$ is the weighted average value of distance measures in the vector spaces corresponding to the available

activity property types. Coefficient w_t is the weight of $dist_t$ indicating the impact of the activity distance according to property type t. We reference all the weights in Equation 1 as $\vec{W} = (w_{t_1}, \ldots, w_{t_n})$, where $n = |T|$. In the next section we will explain the role of vector $\vec{W}$.

4. Process Model Collection Abstraction Fingerprint

The application of different abstraction operations to one process model leads to various abstract representations of the modeled business process. The differences between abstraction operations are explained by their pragmatics, i.e., various abstraction purposes. If the abstraction is realized by a human, the modeling habits of the designer are reflected in the abstraction operation as well. Hence, abstraction pragmatics and modeling habits of the designer are inherent properties of the abstraction operation and together form an *abstraction style*. This section explains how to use vector $\vec{W}$ in Equation 1 to capture an abstraction style.

From the user perspective, vector $\vec{W}$ is the tool to express the desired abstraction style. We can imagine two scenarios on how vector $\vec{W}$ can be obtained. In the first scenario, the user explicitly specifies $\vec{W}$. This approach is useful if the user wants to introduce a new abstraction style. However, coming up with an appropriate value for $\vec{W}$ may be challenging. Hence, the second scenario implies that vector $\vec{W}$ is mined from a process model collection that is enriched with a subprocess relation (formalized with σ in Definition 3). The discovered vector can be seen as a "fingerprint" of the process model collection with respect to the used abstraction style. We will now describe an approach to discover vector $\vec{W}$ from such a process model collection.

The discovery of a model collection's abstraction fingerprint is driven by the following argumentation. Activities of a process model collection are aggregated into more abstract activities, i.e., subprocess placeholders, by the model designer. We aim to achieve an activity clustering algorithm that approximates this aggregation behavior by a human. This is possible if an activity distance measure employed by the algorithm resembles the criteria that a human designer uses to aggregate activities into a subprocess. In general, these criteria are not exactly known. Yet, for each pair of activities we can observe the outcome: Either the activities belong to the same subprocesses or to different ones. For a process model collection $c = (M, A, P, \sigma)$ function *diff* formalizes this observation:

$$diff(a_k, a_l) = \begin{cases} 0, & \text{if } a_k, a_l \in A_i; \\ 1, & \text{otherwise.} \end{cases} \tag{2}$$

To mine the process model collection fingerprint $\vec{W}$ we select its value in such a way that the behavior of function $dist_{agg}$ approximates the behavior of *diff*. The discovery of vector $\vec{W}$ is realized by means of linear regression. In our setting, the values $dist_t$ are considered as independent variables and the value of function *diff* as the dependent variable; the components of vector $\vec{W}$ are the

regression coefficients. The standardized coefficients indicate the impact of each activity property type on the abstraction style. This is a way to approximate the criteria that are employed by the human designer during abstraction decisions. Furthermore, the regression's coefficient of determination R^2 allows to judge how well the obtained statistical model explains the observed behavior. For our purposes, a high R^2 suggests that the discovered statistical model can indeed be used for business process model abstraction.

5. Empirical Validation

The proposed activity aggregation mechanism calls for validation. The goal of the validation is to learn how well the proposed activity aggregation methods approximate the abstraction style of human modelers. First, we evaluate the quality of activity aggregation delivered by the clustering approach. Second, we check whether a process model collection's abstraction fingerprint can be used to capture the abstraction style of model designers. We empirically validate the approaches by empirically exploring a real world business process model collection. This section provides a detailed discussion of the validation; it describes in detail the used process model collection, explains the experiment design, and discusses the validation results.

5.1. Experiment Setting

As a research object we chose a set of business process models from a large telecommunication service provider. This organization is currently in the process of setting up a repository with high-quality process models, which are brought together for the purpose of consultation and re-use by business users. The model set includes 30 elaborate models, enriched with activity properties of the following types: *roles*, *responsible roles*, *IT systems* and *data objects*. In addition, a special type of roles, i.e., *responsible roles*, is also distinguished within these models. Next to these non-control flow types of information, we also study the impact that the activity labels and the proximity between model elements with respect to the control flow relation have on the decision to aggregate activities into the same subprocess. To compare activities with respect to their labels, the corresponding vector space is formed by the words that appear in the labels. Against this background, finding the distance between activities becomes an information retrieval task since labels can be treated as documents in information retrieval. The comparison of activities with respect to their control flow proximity shows whether the neighborhoods of two activities intersect, i.e., contain the same flow elements. Table 2 outlines the relevant properties of the process models. In the existing repository, the models are hierarchically organized using a subprocess relation. Within the model set, we have identified 8 subprocess hierarchies. Each hierarchy contains a root process model refined with subprocesses, allowing for several hierarchical levels of refinement.

	Nodes	Activities	Role	Responsible role	IT system	Data object
Average	15.5	6.3	2.1	0.76	1.5	0.76
Minimum	5	1	0	0	0	0
Maximum	48	20	5	2	7	17

Table 2: Properties of business process models used in the validation

5.2. Similarity Measure Validation

Section 3 has demonstrated how cluster analysis can be used for activity aggregation. However, the practical utility of the proposed solutions has not been determined. This section validates the usefulness of the designed activity aggregation based on K-means clustering. Section 5.2.1 describes the setting of the empirical validation, while the observed results are presented in Section 5.2.2.

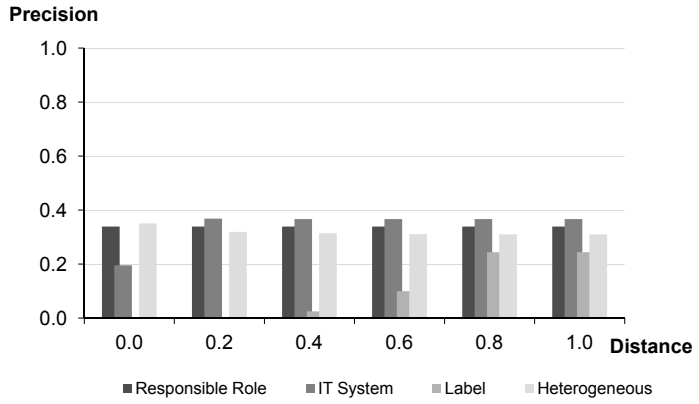
5.2.1. Validation Setup

To evaluate the usefulness of the advocated cluster analysis method we compare the automatically delivered activity aggregations (retrieved activity aggregations) with the aggregations specified by humans (relevant activity aggregations). As quantitative measures, we use the notions of precision, recall, and F-score [3]. *Precision* indicates the share of retrieved aggregations that are relevant, while *recall* is the fraction of relevant aggregations that are retrieved. An *F-score* is the harmonic mean of the precision and the recall.

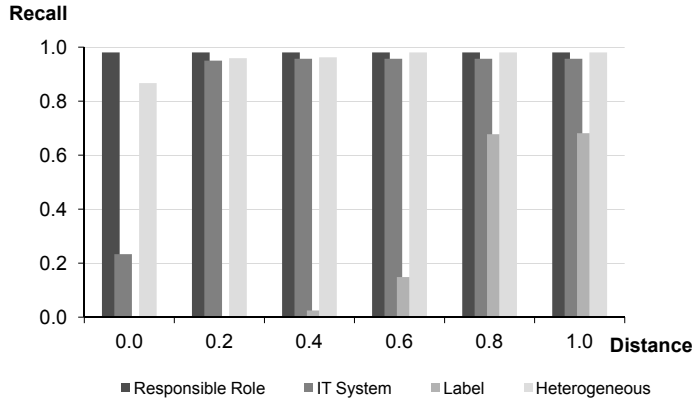
Precision, recall, and F-score allow for the comparison of retrieved and relevant activity aggregations. The reader may note that activity aggregations differ in sizes, often containing more than two activities. This fact complicates the comparison of activity aggregations. Consider an example of two activity aggregations, one with a size of five and the other with a size of six. The two aggregations may actually share four common activities. If we compare the two aggregations as sets, we learn that the aggregations are unequal. Yet, the two activity aggregations are highly similar, as they have a relatively large number of activities in common. To consider the intersection of activity aggregations, we compare activity pairs only. Given an activity aggregation we decompose it into a set of all its subsets of size two. In this context, the comparison of two activity aggregations turns into the comparison of two sets of activity pairs. Consider, for instance, the set $\{a, b, c\}$, where activity c is weakly related to a and b. This set can be decomposed into pairs $\{a, b\}$, $\{a, c\}$, $\{b, c\}$. The fact that c is weakly related to a and b can be easily pointed out by claiming (a, c) and (b, c) to be irrelevant. Against this background, we evaluate the similarity of two activity aggregation sets by comparing each pair from the first activity pair set with each pair in the second activity pair set. We claim that two pairs coincide if they contain the same elements.

5.2.2. Observed Results

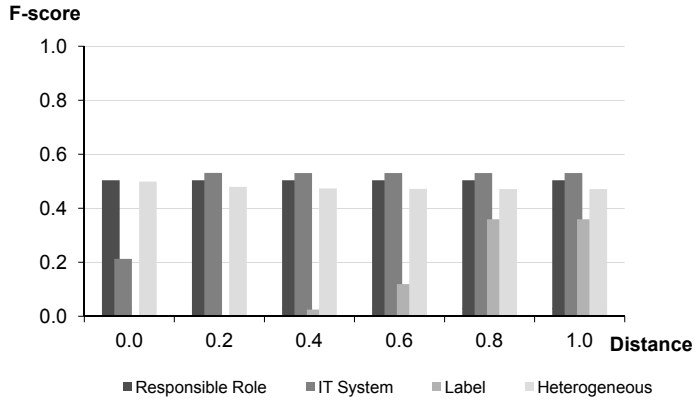
In our validation, we have varied two parameters: the distance value and the vector space type. The distance value varies between 0 and 1 with incremental



(a) Variation of activity aggregation precision with respect to the distance



(b) Variation of activity aggregation recall with respect to the distance



(c) Variation of activity aggregation F-score with respect to the distance

Figure 5: Precision, recall, and F-score observed within the validation of activity aggregation

steps of 0.2. As the vector space types, we have explored homogeneous spaces with the types *Responsible role*, *IT system*, and *Label*, as well as the heterogeneous vector space. The obtained precision, recall, and F-score values are plotted in Figure 5. Note that Figure 5 does not consider the spaces with activity property types *Role* and *Data object* since the observed precision, recall, and F-score values fluctuate around zero in these spaces. The shown precision value varies between 0.00 and 0.37, the recall is between 0.00 and 0.98, and the F-score varies between 0.00 and 0.53. We can make several observations with respect to the obtained data. First, we notice that the values moderately depend on the distance value. This observation does not align with a theoretical assumption that precision, recall, and F-score vary with the distance increase. The graphs vividly illustrate the large difference in the performance of clustering in the homogeneous vector space with the type *label* and the rest vector spaces. To summarize the evaluation of the activity aggregation we conclude with the key observations:

Recall vs. precision The recall dominates the precision.

F-score Activity aggregation exhibits high consistency fluctuating around the value of 0.50.

Possible applications If the application context implies that the recall is more significant, activity aggregation based on clustering is preferable. If the application requires high precision, meronymy-based aggregation outperforms aggregation based on clustering [35]. Finally, the obtained precision, recall, and F-score values witness that the developed aggregation method cannot be used in fully automatic fashion. Against this background, we propose to use the method to support the user with suggestions on activity aggregations.

5.3. Abstraction Fingerprint Validation

Section 4 proposed a method to discover process model abstraction fingerprint in a collection. At this point we provide an empirical evidence illustrating the possibility of abstraction fingerprint extraction. Applying the method to an industrial process model set we discuss the obtained results. Section 5.3.1 briefly presents the validation setup, while Section 5.3.2 elaborates on the observed results.

5.3.1. Validation Setup

To formally validate how well the designed activity aggregation approximates the behavior of modelers clustering a set of activities into the same subprocess, we came up with the following approach. For each pair of activities that belong to the same process hierarchy, we have evaluated two values in the process model collection: *diff* and *dist*. Here, *diff* describes the human abstraction style, which indicates whether the activities have been placed in the same subprocess or not. The value of *dist* represents the vector space distance between the two

Experiment	$\rho(dist_t, diff)$						$\rho(dist_h, diff)$	$\rho(dist_{avg}, diff)$	$\rho(dist_{agg}, diff)$
	Role	Responsible role	IT system	Data object	Label	Neighbor			
All models	0.70	0.61	0.60	—	0.34	0.58	0.74	0.65	0.77
Test_1	0.79	0.76	0.75	—	0.42	0.60	0.79	0.69	0.79
Test_2	0.64	0.56	0.56	—	0.43	0.62	0.68	0.70	0.70
Test_3	0.68	0.58	0.58	—	0.53	0.64	0.68	0.72	0.71
Test_4	0.61	0.47	0.45	—	0.20	0.48	0.70	0.56	0.52
Average_{1-4}	0.68	0.59	0.58	—	0.39	0.59	0.71	0.67	0.68

Table 3: Correlation values observed in the K-fold cross validation

activities in accordance with our approach. To discover if the two approaches yield similar results, we study the correlation between the two variables. A strong correlation of two variables suggests that *dist* would be a good distance measure in the clustering algorithm. In this case, the inclusion of two activities within the same subprocess is mirrored by a similar, close positioning of the corresponding vectors in the vector space. Given the nature of the observed variables, we employ Spearman's rank correlation coefficient.

In the following, we first investigate the human abstraction style in the model collection as a whole. Then, we verify the results organizing a K-fold cross validation. We partition the model sample into 4 subsamples, i.e., $k = 4$ and perform four tests. In each test, three subsamples are used to discover vector $\vec{W}$, while the fourth subsample is used to evaluate the correlation values between the *diff* and *dist* measures in different vector spaces. In this way, an obviously more reliable insight is developed into the question whether the human abstraction style can be mimicked than by using the whole process model collection for both the discovery and the evaluation of this correlation.

5.3.2. Observed Results

Table 3 outlines the validation's results. The columns in the table correspond to distance measures. While the first 6 columns correspond to distances in homogeneous spaces, the last three columns reflect the distance measure taking into account multiple activity properties. All three distance measures make use of the activity property types in columns 1–6. The distance $dist_h$ is measured in heterogeneous vector space, where dimensions are activity property values of types listed in columns 1–6. The distance measure $dist_{avg}$ is the average value of distances in columns 1–6. The distance measure $dist_{agg}$ is evaluated according to Equation 1. Vector $\vec{W}$ used in $dist_{agg}$ is obtained using linear regression as described in the previous section. Rows of Table 3 correspond to experiments. The first row describes the study of the whole model collection. Rows 2–5 describe the results of 4 tests along the K-fold cross validation we

explained earlier, while the last row provides the average correlations observed in the 4 separate tests.

The correlation values that are presented in Table 3 are all significant using a confidence level of 99%, i.e., all p values are lower than 0.01. However, no statistically significant results were obtained for the distance in the homogeneous vector space that corresponds to *Data objects*. Overall, the presented correlation values range around 0.7. This level is generally considered to indicate a strong correlation [12, 13], even more so in situations where human decision making is involved. Therefore, we can speak of a strong relation between the *dist* and *diff* measures.

Among the distance measures in homogeneous spaces, one can point out the distance in the *Role* space that overall displays the highest correlation values for the different studies (0.61–0.79). In contrast, correlation values for *Label* are the lowest (0.20–0.53). Another observation is that distances taking into account multiple activity property types tend to have higher correlations. From these, $dist_{agg}$ outperforms all other distance measures with a value arriving at 0.77 when all models are considered. For the average values of the K-fold cross validations, however, $dist_h$, $dist_{avg}$, and $dist_{agg}$ demonstrate a similar performance, with correlation values of 0.71, 0.67, and 0.68 respectively. This observation can be explained by the fact that $dist_{agg}$ is parameterized by vector $\vec{W}$, the abstraction fingerprint of a particular model set. Thus, the distance measure $dist_{agg}$ "trained" on one model set may never excel $dist_{avg}$, once the set of models is changed. Tests 1–4 support this argumentation. Note that this result does not restrict the applicability of the approach: In a real world setting, the goal is to transfer the abstraction style from one model set to another. The average values in the lower row should, therefore, be seen as most important among the ones displayed.

A careful inspection of the linear regression results associated with parameterizing vector $\vec{W}$ provides additional insights. In particular, we are interested in the observed R^2 values and the beta coefficients (also known as 'standardized coefficients'). The R^2 for the whole model set, as well as the average value for the K-fold cross validation is 0.52. This value shows the explained level of variation in abstraction style as explained by the various distance measures under consideration and can be considered as moderately strong. The beta coefficients of the distance measures in various spaces reveal their impact on the activity aggregation. The beta coefficients for activity property types *Role* and *Responsible role* have average values of 0.55 and 0.37, respectively. At the same time, the standardized coefficients of *Neighbor* and *Label* property types fluctuate around 0. The average value for *IT systems* is somewhere in between, with a beta coefficient of 0.19. The provided numbers illustrate that the activity property types *Role* and *Responsible role* have a big impact on the abstraction style of the considered process model collection. *IT systems* does contribute to the activity aggregation as well, but the influence of activity labels and activity neighborhood is insignificant. Clearly, such insights may differ from one process model collection to the other.

The validation indicates that the suggested distance measures can be used to closely approximate the abstraction style of human modelers. Among the introduced measures, $dist_{agg}$ is of particular interest as it takes into account the abstraction style of a specific process model collection. Furthermore, the empirical validation revealed activity property types, *Role* and *Responsible role*, that have the highest impact on the abstraction style for this particular collection.

6. Related Work

The topic of business process model abstraction can be related to several research streams. We discuss these streams looking both from the perspective of the software engineering and business process management disciplines.

Model properties and relations are thoroughly investigated in the software engineering area. For instance, in [22, 23] Kühne elaborates on the concepts of model, metamodel, model types, and model relations. These works systematically describe and organize relations, e.g., generalization and classification, which are seminal to the problem of model abstraction. Closely related are also the studies that cover model granularity. In [18], the authors investigate model and metamodel granularity. The authors compare several metamodels and come up with best practices with respect to granularity. One can observe that the relation between a coarse-grained activity in an abstract model and its counterparts in the initial model is the meronymy, or part-of, relation. Meronymy has been studied in depth in the software engineering domain [4, 14]. Although the referenced papers do not provide concrete techniques for the implementation of abstraction within process models, they facilitate a better problem understanding and help to identify the main concepts in this domain.

Business process management is the discipline concerned with using methods, techniques, and software to design, enact, control, and analyze operational processes. A large body of knowledge corresponds to the topic of process model analysis based on model transformations. Model transformations can be reused in the context of the abstraction problem. An example of such a transformation consists of reduction rule sets for Petri nets, e.g., see [6, 26, 31]. Each reduction rule explicitly defines a structural fragment to be discovered in the model and a method of this fragment transformation. Hence, reduction rule sets enable process model abstraction through an iterative application of rules. As the transformed process fragments are explicitly defined, each reduction rule set handles only a particular model class. Thereby, each reduction rule set requires an argument about the model class reducible with the given rules. The model class limits the application of abstraction approaches based on reduction rules [7, 11]. Process model decomposition approaches are free of this limitation: They seek for process fragments with particular properties. An example of such a decomposition is presented in [38], where single entry single exit fragments are discovered. The result of process model decomposition is the hierarchy of process fragments according to the containment relation, i.e., the process structure tree. Such a tree can indeed be used for abstraction in process models [28]. Finally, one can distinguish model transformations that preserve process behavior

properties. In [1], van der Aalst and Basten introduce three notions of behavioral inheritance for WF-nets and study inheritance properties. The paper suggests model transformations, such that the resulting model inherits the behavior of the initial model. An approach for process model abstraction can exploit such transformations as basic operations. While the outlined model transformations can support solving the general problem of process model abstraction, they all focus on structural and behavioral aspects of models and model transformations, leaving the semantic aspect out of scope.

Many tasks in the management of large process model collections can be traced back to the problem of *activity matching*, which is closely related to the problem of business process model abstraction. Examples of such management tasks are: the search for a particular process model over a process model set or ensuring the consistency of models capturing one and the same process from different perspectives. Activity matching is realized through analysis of activity properties: activity labels, referenced data objects and neighboring activities. In [10, 39] the authors suggest activity matching algorithms and evaluate them. While the named works explore the existing process models and do not directly address the problem of process model abstraction, their results have a potential of being applied in business process model abstraction. Semantic aggregation of activities relates to research on semantic business process management. Notice that process models enriched with semantic information facilitate many process analysis tasks, see [19]. Along this line of research, several authors argue how to use activity ontologies to realize activity aggregation [8, 25]. It should be noticed, however, that such works imply the existence of a semantic description for model elements and their relations, which is a restriction that rarely holds in real world settings.

Establishing an activity's granularity level is also a recurrent challenge in process mining, where logs contain records that are often very fine-granular. As such, the process models directly mined from the logs can be overloaded with information making them hard to comprehend. Activity clustering is an efficient means to raise the abstraction level for the mined models. In [15, 16] Günther and van der Aalst propose activity aggregation mechanisms based on clustering algorithms. The mechanisms extensively use information present in process logs, but which are less common for process models, i.e., timestamps of activity starts and stops, activity frequencies, and transition probabilities. Thus, in contrast to the activity aggregation approach proposed in this paper, process mining considers other activity property types for clustering and utilize other clustering algorithms.

7. Conclusions and Future Work

Despite the fact that the topic of business process model abstraction has been addressed in a number of research endeavors, this paper proposes a novel approach in this area. Specifically, it exploits semantic aspects—beyond the control-flow perspective—to determine the similarity between different activities for the purpose of abstracting from overly complex process models. As shown,

the relevant levels for evaluating such similarities can be determined on the basis of an existing collection of process models to which abstraction has already been applied.

Our main contribution is a method to discover sets of related activities, where each set corresponds to a coarse-grained activity in an abstract process model. As a second contribution, we propose an approach to discover the abstraction style that is inherent to a given process model collection, which is reusable for new abstraction operations. Both contributions are of high practical interest since they addresses model management issues recurrently appearing in process modeling projects. The experimental validation provides strong support for the applicability and effectiveness of the presented ideas.

Our approach is characterized by a number of limitations and assumptions. First of all, we build on the assumption that all kinds of semantic information, such as data objects, roles, and resources, can be observed within the descriptions of process models in industrial collections. The process model collection we obtained through our cooperation with a large telecommunication company indeed confirms this idea, just as this seems to be the case for other industrial repositories that are often referenced, e.g., the SAP Reference Model [21]. Clearly, we cannot make a universal claim about the validity of our assumption. Secondly, in our validation we have merely focused on the appearance—or lack thereof—of two activities within the same subprocess or process model. It can certainly be imagined that a more fine-grained correspondence measure would yield different and perhaps even more useful results.

These and other limitations guide our future research plans. The direct next step for us is to consider other clustering algorithms and compare their results with the solution introduced in this paper. From a more practical perspective, it is important to come up with suggestions for names of coarse-grained activities that form the product of activity aggregation. Finally, we would like to improve the validation method for activity aggregation. In particular, a validation that would involve human modelers and stakeholders who can provide qualified feedback would further strengthen the confidence in and practical use of the proposed activity aggregation approach.

References

[1] W. M. P. van der Aalst and T. Basten. Life-Cycle Inheritance: A Petri-Net-Based Approach. In *ICATPN 1997*, volume 1248 of *LNCS*, pages 62–81. Springer, 1997.

[2] E.R. Aguilar, F. Ruiz, F. García, and M. Piattini. Evaluation Measures for Business Process Models. In *2006 ACM Symposium on Applied Computing*, pages 1567–1568. ACM, 2006.

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

[4] 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 Transactions on Software Engineering*, 29(5):459–470, 2003.

[5] J. Becker and D. Kuropka. Topic-based Vector Space Model. In *BIS 2003*, pages 7–12. GI, 2003.

[6] G. Berthelot. Transformations and Decompositions of Nets. In *Advances in Petri nets 1986*, pages 359–376. Springer, 1987.

[7] R. Bobrik, M. Reichert, and T. Bauer. View-Based Process Visualization. In *BPM 2007*, volume 4714 of *LNCS*, pages 88–95. Springer, 2007.

[8] F. Casati and M.-Ch. Shan. Semantic Analysis of Business Process Executions. In *EDBT 2002*, pages 287–296. Springer, 2002.

[9] C. Di Francescomarino, A. Marchetto, and P. Tonella. Cluster-based Modularization of Processes Recovered from Web Applications. *Journal of Software Maintenance and Evolution: Research and Practice*, 2010.

[10] R. M. Dijkman, M. Dumas, and L. García-Bañuelos. Graph Matching Algorithms for Business Process Model Similarity Search. In *BPM 2009*, volume 5701 of *LNCS*, pages 48–63. Springer, 2009.

[11] R. Eshuis and P. Grefen. Constructing Customized Process Views. *Data & Knowledge Engineering*, 64(2):419–438, 2008.

[12] A.N. Franzblau. *A Primer of Statistics for Non-statisticians*. Harcourt, Brace & World New York, 1958.

[13] B. B. Gerstman. StatPrimer – Version 6.4. Technical report, San Jose State University, 2010. `http://www.sjsu.edu/faculty/gerstman/StatPrimer/`.

[14] G. Guizzardi. Modal Aspects of Object Types and Part-Whole Relations and the *de re/de dicto* Distinction. In *CAiSE 2007*, volume 4495 of *LNCS*, pages 5–20. Springer, 2007.

[15] C. W. Günther and W. M. P. van der Aalst. Mining Activity Clusters from Low-Level Event Logs. *BETA Working Paper Series*, WP 165, 2006.

[16] C. W. Günther and W. M. P. van der Aalst. Fuzzy Mining—Adaptive Process Simplification Based on Multi-perspective Metrics. In *BPM 2007*, volume 4714 of *LNCS*, pages 328–343. Springer, 2007.

[17] J. Hartigan. *Clustering Algorithms*. John Wiley and Sons, New York, USA, 1975.

[18] B. Henderson-Sellers and C. Gonzalez-Perez. Granularity in Conceptual Modelling: Application to Metamodels. In *ER 2010*, volume 6412 of *LNCS*, pages 219–232. Springer, 2010.

[19] 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*, pages 535–540. IEEE Computer Society, 2005.

[20] M. Indulska, J. Recker, M. Rosemann, and P. Green. Business Process Modeling: Current Issues and Future Challenges. In *CAiSE 2009*, volume 5565 of *LNCS*, pages 501–514. Springer, 2009.

[21] G. Keller and T. Teufel. *SAP R/3 Process Oriented Implementation*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1998.

[22] Th. Kühne. Matters of (Meta-) Modeling. *Software and Systems Modeling*, 5(4):369–385, 2006.

[23] Th. Kühne. Contrasting Classification with Generalisation. In *APCCM 2009*, volume 96 of *CRPIT*, January 2009.

[24] D. Kuropka. *Modelle zur Repräsentation natürlichsprachlicher Dokumente – Information-Filtering und -Retrieval mit relationalen Datenbanken*. Logos Verlag, Berlin, Germany, 2004.

[25] A.K. Alves 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.

[26] J. Mendling, H. Verbeek, B. van Dongen, W. M. P. van der Aalst, and G. Neumann. Detection and Prediction of Errors in EPCs of the SAP Reference Model. *Data & Knowledge Engineering*, 64(1):312–329, 2008.

[27] OMG. OMG Unified Modeling Language (OMG UML) 2.3, May 2010.

[28] A. Polyvyanyy, S. Smirnov, and M. Weske. The Triconnected Abstraction of Process Models. In *BPM 2009*, volume 5701 of *LNCS*, pages 229–244. Springer, 2009.

[29] H. A. Reijers and J. Mendling. Modularity in Process Models: Review and Effects. In *BPM 2008*, pages 20–35. Springer, 2008.

[30] M. Rosemann. Potential Pitfalls of Process Modeling: Part A. *Business Process Management Journal*, 12(2):249–3254, 2006.

[31] W. Sadiq and M. E. Orlowska. Analyzing Process Models Using Graph Reduction Techniques. *Information Systems*, 25(2):117–134, 2000.

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

[33] S. E. Schaeffer. Graph Clustering. *Computer Science Review*, 1(1):27–64, 2007.

[34] A. Sharp and P. McDermott. *Workflow Modeling: Tools for Process Improvement and Applications Development*. Artech House Publishers, 2008.

[35] S. Smirnov, R. M. Dijkman, J. Mendling, and M. Weske. Meronymy-Based Aggregation of Activities in Business Process Models. In *ER 2010*, volume 6412 of *LNCS*, pages 1–14. Springer, 2010.

[36] S. Smirnov, H. A. Reijers, T. Nugteren, and M. Weske. Business Process Model Abstraction: Theory and Practice. Technical Report 35, Universit "at Potsdam, 2010. `http://bpt.hpi.uni-potsdam.de/pub/Public/SergeySmirnov/abstractionUseCases.pdf`.

[37] I. T. P. Vanderfeesten, H. A. Reijers, J. Mendling, W. M. P. van der Aalst, and J. Cardoso. On a Quest for Good Process Models: The Cross-Connectivity Metric. In *CAiSE 2008*, volume 5074 of *LNCS*, pages 480–494. Springer, 2008.

[38] J. Vanhatalo, H. Völzer, and J. Koehler. The Refined Process Structure Tree. In *BPM 2008*, pages 100–115. Springer, 2008.

[39] M. Weidlich, R. M. Dijkman, and J. Mendling. The ICoP Framework: Identification of Correspondences between Process Models. In *CAiSE 2010*, volume 6051 of *LNCS*, pages 483–498. Springer, 2010.

[40] S. K. M. Wong, W. Ziarko, and P. C. N. Wong. Generalized Vector Spaces Model in Information Retrieval. In *SIGIR*, SIGIR 1985, pages 18–25, New York, NY, USA, 1985. ACM.