Bus Inf Syst Eng 60(6):525–542 (2018)
https://doi.org/10.1007/s12599-016-0457-x

RESEARCH PAPER

Business Process Modeling Abstraction Based on Semi-Supervised Clustering Analysis

Nan Wang · Shanwu Sun · Dantong OuYang

Received: 20 October 2014 / Accepted: 26 June 2016 / Published online: 2 December 2016

Abstract The most prominent Business Process Model Abstraction (BPMA) use case is the construction of the process "quick view" for rapidly comprehending a complex process. Some researchers propose process abstraction methods to aggregate the activities on the basis of their semantic similarity. One important clustering technique used in these methods is traditional *k*-means cluster analysis which so far is an unsupervised process without any priori information, and most of the techniques aggregate the activities only according to business semantics without considering the requirement of an order-preserving model transformation. The paper proposes a BPMA method based on semi-supervised clustering which chooses the initial clusters based on the refined process structure tree and designs constraints by combining the control flow consistency of the process and the semantic similarity of the activities to guide the clustering process. To be more precise, the constraint function is discovered by mining from a process model collection enriched with subprocess relations. The proposed method is validated by applying it to a process model repository in use. In an experimental validation, the proposed method is compared to the traditional k-means clustering (parameterized with randomly chosen initial clusters and an only semantics-based distance measure), showing that the approach closely approximates the decisions of the involved modelers to cluster activities. As such, the paper contributes to the development of modeling support for effective process model abstraction, facilitating the use of business process models in practice.

Keywords Business process model abstraction · Order-preserving · Semi-supervised clustering · Activity aggregation · Constrained k-means clustering · Virtual document

Accepted after three revisions by Prof. Dr. Becker.

N. Wang · S. Sun (✉)
Department of Management Science and Information Engineering, Laboratory of Logistics Industry Economy and Intelligent Logistics, Jilin University of Finance and Economics, Changchun 130117, China
e-mail: ctusunshanwu@126.com

N. Wang
e-mail: ctuwangnan@126.com

D. OuYang
Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China
e-mail: ouyangdantong@163.com

1 Introduction

Scientific papers that describe Business Process Modeling Abstraction (BPMA) techniques by no means always use this exact label, but rather refer to developing process views (see Bobrik et al. 2007a; Eshuis and Grefen 2008), or focus on process simplification (see Günther and van der Aalst 2007). The essential purpose of these techniques is in line with the way BPMA was characterized by Smirnov et al. (2012). In Smirnov et al. (2010b, 2012), the authors show that the most prominent use case of BPMA is a construction of a process "quick view" for rapidly comprehending a complex process. To deal with such a demand, the process model can then be displayed as a partially ordered set of coarse-grained activities, each of which is correlated to a group of lower-level activities. Obviously, there are alternative ways to aggregate activities. From the user perspective, groups of activities that semantically belong together are of particular value (Smirnov 2012). The structure-based abstraction (Polyvyanyy et al. 2008, 2009a; Vanhatalo et al. 2009) derives coarse-grained activities only based on control flow relations but not considering the domain semantics of activities, so that it can not answer such questions as "how to discover the domain interrelated activities". To overcome the disadvantages of the structure-based abstraction, some researchers investigate methods that aggregate activities according to their business meaning. A number of recent contributions exist that consider semantic aspects for aggregation (e.g., Smirnov et al. 2010b; Francescomarino et al. 2013). However, their assumptions, e.g., the existence of an activity ontology (Smirnov et al. 2010b), are too strict for generic use. The approach in Smirnov et al. (2011) is based on the application of the vector space model, an algebraic model popular in information retrieval (Salton et al. 1975). But the space dimensions correspond to activity property values *p* which builds 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. Moreover, these semantics-based methods aggregate the activities only according to business semantics similarity but not considering the order-preserving requirement of the model transformation (Bobrik et al. 2007a; Eshuis and Grefen 2008; Smirnov 2012; Polyvyanyy et al. 2009a; Liu and Shen 2003), so that the activities in the generated clusters (candidate subprocesses) are relatively dispersed from the perspective of order consistency and structural connectedness (Reijers et al. 2010).

Therefore, this paper investigates methods that aggregate activities according to both their business semantics and control flow consistency. In other words, given a business process model, we search for activity sets that each has a self-contained business semantics with as little as possible control flow loss. As an example, a process model (Smirnov et al. 2011) that captures the creation of a forecasting report is shown in Fig. 1. We assume that there are four reasonable subprocess candidates, and different shadings are used to mark the corresponding activities.

In this paper, we propose a BPMA method based on a semi-supervised clustering algorithm. Semi-supervised clustering is based on unsupervised clustering, e.g., the *k*-means clustering used in (Smirnov et al. 2011), by using labeling data (or constraint relations) to guide the clustering process in order to improve the quality of clustering (Gao et al. 2008). Semi-supervised clustering algorithms can be divided into three categories: the first contains a constraint-based semi-supervised clustering algorithm which uses the class labels or pairwise constraints to improve clustering algorithm itself (Wagstaff and Cardie 2000; Wagstaff et al. 2001; Basu et al. 2002, 2004; Demiriz et al. 1999; Bilenko et al. 2004; Ruiz et al. 2007; Gaynor and Bair 2013). The second category includes metric-based or distance-based semi-supervised clustering algorithms. Such algorithms use the class labels or pairwise constraints to learn a new distance measure function to satisfy the constraints (Kamvar et al. 2003; Xu et al. 2005; Klein et al. 2002; Wang et al. 2007; Xing et al. 2003; Schultz and Joachims 2003; Bar-Hillel et al. 2003; Tang et al. 2007; Hastie et al. 2009; Cohn et al. 2009; Yin et al. 2010). The third method is a combination of these two kinds of semi-supervised clustering algorithms (Bilenko et al. 2004; Basu et al. 2004; Tang et al. 2007). Our approach designs the initial parameters and constraints by combining the control flow consistency of the process and the semantic similarity of the activities to guide the clustering process. Due to the order-preserving and block-structured nature of the abstraction based on the refined process structure tree (RPST) decomposition (Vanhatalo et al. 2009; Reijers et al. 2010), we choose *k* canonical components of the RPST constructed for the process model as the initial clusters (seed sets) and compute the initial cluster centroids (centers). We also design a new constraint function by combining the semantic similarity and the control flow ordering requirement to constrain the traditional *k*-means clustering process, not only aggregating the activities with similar business semantics but also reducing the control flow loss of the abstract results. In particular, we discover the constraint function by mining from a process model collection enriched with subprocess relations. We validate the proposed method by applying it to a process model repository

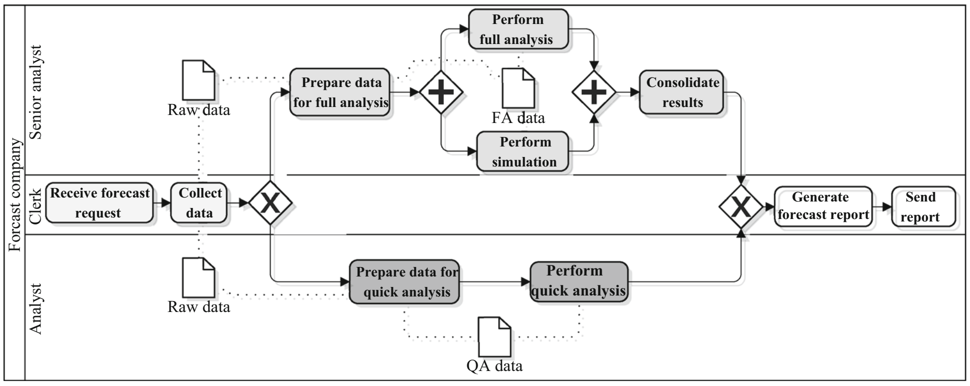

Fig. 1 Motivating example: initial model and its activity groups to be abstracted

that is in use with a large joint venture automobile production enterprise (China's largest automobile manufacturing organization) and its partner logistics company. The repository incorporates hierarchical relations between high-level activities and the activities that they aggregate. Also, the process models contain various types of semantic information. In an experimental validation, we compare the proposed method to the traditional k-means clustering (parameterized with randomly chosen initial clusters and a distance measure solely based on semantics), showing that our approach closely approximates the decisions of the involved modelers to cluster activities.

The structure of the paper is as follows. In Sect. 2 we continue explaining the proposed algorithm, along with providing the required background knowledge. Section 3 empirically validates the proposed approach by using an industrial set of process models from the companies we mentioned above. Finally, Sect. 4 concludes the paper with a summary and discussion.

2 Activity Aggregation

This section elaborates on the proposed activity aggregation algorithm. After the introduction of the main concepts, we argue how activity aggregation can be interpreted as a semi-supervised clustering problem. We discuss a constrained clustering algorithm with suitable initial parameters. We explain how the aggregation setup is realized and show how the setup information can be mined from an existing process model collection.

2.1 Fundamentals

When business users talk about process model abstraction, they often imply the abstraction of activities, requesting a transition from low level steps to high level tasks (Polyvyanyy et al. 2009a). In this section, we introduce some concepts of Smirnov (2012) which will be used in the subsequent sections.

2.1.1 Process Model Decomposition

Firstly, we introduce Smirnov's concept of the business process model (Smirnov 2012).

Definition 2.1 A tuple PM = (A, G, F, t, s, e) is a business process model where:

- A is a finite nonempty set of activities.
- G is a finite set of gateways.
- $N = A \cup G$ is a finite set of nodes with $A \cap G = \varnothing$.
- $F \subseteq N \times N$ is the flow relation, such that (N, F) is a connected graph.
- Every activity has no more than one incoming and no more than one outgoing edge.
- s is the only one activity which has no incoming edges – a start activity and e is the only one activity which has no outgoing edges – an end activity.
- $t : G \rightarrow \{and, xor\}$ is a function that assigns to each gateway a control flow construct.
- Every gateway is either a split or a join; splits have exactly one incoming edge and at least two outgoing ones; joins have at least two incoming edges and exactly one outgoing one.

The execution semantics of a process model is given by a translation into a Petri net following common formalizations (see Smirnov 2012 in detail). Then, we introduce the decomposition into fragments with single entry nodes and single exit nodes which results in a RPST. According to Smirnov (2012), in the context of process modeling the resulting fragments can be considered as self-contained process parts. As such fragments have a single entry node and a single exit node, structurally they can be isolated into a subprocess. And this decomposition is unique. The RPST can be constructed in time linear to the number of nodes in the process model (Polyvyanyy et al. 2010).

Definition 2.2 Let PM = (A, G, F, t, s, e) be a process model. A *fragment* f of process model PM is a tuple f = (A_f, G_f, F_f, t_f) where $\left(A_f \cup G_f, F_f\right)$ is the connected subgraph of the graph $(A \cup G, F)$ and function t_f is the restriction of t of PM to set to G_f.

Definition 2.3 Let PM = (A, G, F, t, s, e) be a process model with a process model fragment PMF = (A_{PMF}, G_{PMF}, F_{PMF}, t_{PMF}). A node $n \in N_{\mathrm{PMF}}$ is a *boundary* node of PMF if $\exists e \in in(n) \cup out(n)$, in which the functions $in(n)$ and $out(n)$ are respectively the sets of the outgoing edges and incoming edges of node n. If n is a boundary node, it is an entry of PMF, if $in(n) \cap F_{\mathrm{PMF}} = \emptyset$. A node n is an exit of PMF, if it is a boundary node of PMF and $out(n) \cap F_{\mathrm{PMF}} = \emptyset$.

Definition 2.4 Let PM = (A, G, F, t, s, e) be a process model with a process model fragment PMF = (A_{PMF}, G_{PMF}, F_{PMF}, t_{PMF}). The fragment PMF is a *component* if it has exactly two boundary nodes: one entry node and one exit node.

Let F be the set of all components in a process model PM.

Definition 2.5 A component PMF = (A_{PMF}, G_{PMF}, F_{PMF}, t_{PMF}) is *canonical* if $\forall \mathrm{PMF}' \in F : \mathrm{PMF} \neq \mathrm{PMF}' \Rightarrow (F_{\mathrm{PMF}} \cap F_{\mathrm{PMF}'} = \emptyset V (F_{\mathrm{PMF}} \subset F_{\mathrm{PMF}'}) \vee (F_{\mathrm{PMF}'} \subset F_{\mathrm{PMF}}))$.

Definition 2.6 Let PM = (A, G, F, t, s, e) be a process model. The *refined process structure tree* of a process

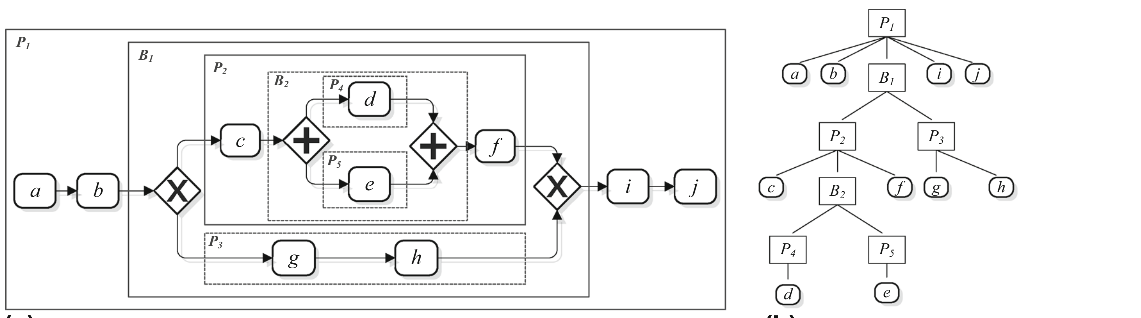
(a) A process model decomposed into canonical component
(b) A RPST of the process model in (a)

Fig. 2 A simplified process model of Fig. 1 and its RPST

model PM is an arborescence $\mathrm{RPST}_{\mathrm{PM}} = (\Omega, r, \chi)$ such that:

- Ω is a set of all canonical components of PM.
- r is a component that is the root of the tree.
- $\chi \subseteq \Omega \times \Omega$ is a relation between a component and its child component.

We take the simplified version of Fig. 1 for example to show the structural decomposition, see Fig. 2.

2.1.2 *Order-Preserving Abstraction*

We introduce the order-preserving abstraction defined in Smirnov (2012) according to the behavior of systems which can be described in terms of behavioral profiles (Weidlich et al. 2011). Let $\mathcal{T}_{PM}$ be the set of complete process traces for a process model PM which contains lists of the form $sA * e$ such that a list comprises the execution order of activities. Let $a \in \sigma$ with $\sigma \in \mathcal{T}_{PM}$ denote that an activity a is a part of a complete process trace.

A behavioral profile captures behavioral characteristics of a process model by three relations between pairs of activity nodes. These relations are based on the notion of *weak order*. Two activities of a process model are in weak order, if a trace exists in which one activity occurs after the other.

Definition 2.7 (*Weak Order Relation*) Let $\mathrm{PM} = (A, G, F, t, s, e)$ be a process model, and $\mathcal{T}_{PM}$ its set of traces. The weak order relation $\succ_{PM} \subseteq (A \times A)$ contains all pairs (a, b), such that there is a trace $\sigma = n_1, \ldots, n_l$ in $\mathcal{T}_{PM}$ with $j \in \{1, \ldots, l-1\}$ and $j < k \leq l$ for which holds $n_j = a$ and $n_k = b$.

Based on the weak order relation, the behavioral profile is defined as follows.

Definition 2.8 (*Behavioral Profile*) Let $\mathrm{PM} = (A, G, F, t, s, e)$ be a process model. A pair $(a, b) \in (A \times A)$ is in one of the following relations:

- strict order relation $\rightsquigarrow_{PM}$, if $a \succ_{PM} b$ and $a \not\succ_{PM} b$.
- Exclusiveness relation $+_{PM}$, if $a \not\succ_{PM} b$ and $b \not\succ_{PM} a$.
- Interleaving order relation $||_{PM}$, if $a \succ_{PM} b$ and $b \succ_{PM} a$.

The set of all three relations $BP = \{\rightsquigarrow_{PM}, +_{PM}, ||_{PM}\}$ is the behavioral profile of PM. $a \not\succsim_{PM} b$ represent there is no weak order relation from a to b. The relations of the behavioral profile, along with the inverse strict order $\rightsquigarrow^{-1} = \{(a, b) \in (A \times A) | (b, a) \in \rightsquigarrow\}$, partition the Cartesian product of activities.

Definition 2.9 (*Function Aggregate*) Let $\mathrm{PM} = (A, G, F, t, s, e)$ be a process model and $\mathrm{PM}_a = (A_a, G_a, F_a, t_a, s_a, e_a)$ its abstract counterpart. Function aggregate: $A_a \rightarrow (P(A) \backslash \emptyset)$ specifies a correspondence between one activity in PM_a and the set of activities in PM.

Definition 2.10 (*Order-Preserving Business Process Model Abstraction*) Let $\mathrm{PM} = (A, G, F, t, s, e)$ be a process model, and business process model abstraction α maps PM to $\mathrm{PM}_a = (A_a, G_a, F_a, t_a, s_a, e_a)$, i.e., $\alpha : (\mathrm{PM}, \textit{activity groups}) \rightarrow \mathrm{PM}_a$, so that activities of *PM* are abstraction objects. Let the function aggregate also establish a correspondence between activities of PM and PM_a. Operation α is order-preserving business process model abstraction, iff $\forall x, y \in A_a, x \neq y$ holds that $\forall a, b \in A$ such that $a \in \textit{aggregate}(x)$ and $b \in \textit{aggregate}(y)$:

- $a \rightsquigarrow_{PM} b \Rightarrow x \rightsquigarrow_{PM_a} y$
- $a \rightsquigarrow^{-1}_{PM} b \Rightarrow x \rightsquigarrow^{-1}_{PM_a} y$
- $a +_{PM} b \Rightarrow x +_{PM_a} y$
- $a ||_{PM} b \Rightarrow x ||_{PM_a} y$.

2.2 Activity Aggregation as a Semi-Supervised Cluster Analysis Problem

Activity aggregation can be conducted according to some structural criteria, such as pattern-based methods (Polyvyanyy et al. 2008; van der Aalst et al. 2003; Gschwind et al. 2008; Smirnov et al. 2009) and decomposition-based methods (Polyvyanyy et al. 2009a, b; Vanhatalo et al. 2007, 2009). The process fragments discovered by structural methods of BPMA are not always semantically complete and the activities contained in the discovered process fragments may not semantically belong together (here we refer to business semantics). Activity aggregation can also be interpreted as a problem of cluster analysis (Smirnov et al. 2011) according to the business semantics of activity. The set of objects to be clustered is the set of activities A_i. The objects are clustered based on a distance measure: objects that are "close" to each other according to this measure are put together. The semantic part of the distance measure can be computed according to various representations for the business semantics of activities. To avoid that the strong assumption from the space dimensions corresponds to activity property values (Smirnov et al. 2011) or that the cluster error results from insufficient information described only by an activity label (Reijers et al. 2010), we consider utilizing as much information related to the activity as possible by introducing the virtual document (Qu et al. 2006), as Weidlich et al. (2010) did, to represent an activity. In addition, we interpret activity aggregation as a problem of semi-supervised cluster analysis and consider not only the business semantics similarity of activities but also the requirement of an order-preserving model transformation.

We now proceed to a discussion of our modifications to the traditional k-means clustering of BPMA used by Smirnov et al. (2011). Firstly, instead of a random method, we choose initial clusters (seed sets) by using the canonical components of the RPST, which, according to the block-structuredness of the subprocess (Reijers et al. 2010), can be a good basis for detecting subprocesses. Thus the knowledge that combines the control flow consistency and the semantic similarity between the grouped activities can be expressed as a set of instance-level constraints on the clustering process. After a discussion of the kind of constraints we are using, we describe the constrained k-means clustering algorithm of BPMA.

The final issue is how to choose k. For the process model that is flattened from a model with human-designed subprocesses (used for our empirical validation in Sect. 3), we make use of the value of k where it is already known (i.e., all of the manually designed subprocesses); for the practical problem of finding subprocesses in a flattened business process model with unknown k, we use a value of k specified by users according to their experience. 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 (Sharp and McDermott 2008). Provided a fixed number, e.g., 6, the clustering algorithm has to assure that the number of clusters equals the request by the user. In addition, in future work we will design a proper evaluation index to assess the process abstraction models and generate the optimal number of the subprocesses.

2.3 Constrained k-Means Clustering of BPMA

2.3.1 Initial Clusters

The requirement for a business process to be block-structured is quite common (Reijers et al. 2010). The canonical components of the RPST are a good basis for detecting subprocesses, i.e., the activities of the same canonical component tend to belong to the same subprocess. Therefore, under the presumption that the original process model is block-structured, we choose k canonical components of the RPST constructed for the process model in question as the initial clusters. Before generating the initial clusters we make the following assumptions in order to maximize the dispersion while choosing the initial clusters in the process model PM. These assumptions restrict the left-to-right order of the nodes of each hierarchy in RPST, i.e., for any pair of son nodes, y and z, of node x:

– y is on the left of z, if $y \rightsquigarrow_{\mathrm{PM}} z$,
– z is on the left of y, if $y \rightsquigarrow_{\mathrm{PM}}^{-1} z$,
– The order of y and z is random, otherwise.

Let T be the RPST of a process model PM, and let's call the canonical component that is composed of a single activity the *atomic component*. We choose k initial clusters (seed sets) into the set $S = (S_1, \ldots, S_k)$ successively in the following order of priority:

1. For each canonical component C of T, if C is composed of more than one individual activities or atomic components, then $S_i \leftarrow C$; repeat with $i \leftarrow i + 1$ until no such canonical component is left or $i > k$;

2. If $i < k$, then randomly select one activity from each hierarchy as seed S_i from those leaf nodes of T which are on a different hierarchy than the already chosen nodes; repeat with $i \leftarrow i + 1$ until no such activity is left or $i > k$;
3. If $i < k$, then randomly select one activity not directly adjacent to the already chosen nodes as seed S_i; repeat with $i \leftarrow i + 1$ until no such activity is left or $i > k$;
4. If $i < k$, randomly select one single activity as seed S_i; repeat with $i \leftarrow i + 1$ until $i > k$.

The above steps preferentially choose the canonical components composed of single activities or atomic components as the initial clusters [step (1)]. From the perspective of the control flow's order-preserving requirement, there is a high probability for such components to be contained in a subprocess. Step (2) to step (4) are executed when the number of the initial clusters is less than k; the first two of these steps try to ensure that dispersed single activities are selected, and the last step randomly selects single activities to complete the construction of k initial clusters. The number k of subprocesses (clusters) in a business process model is far smaller than the number of activities; therefore when we use the real world process models to generate initial clusters, we can in most cases generate all the k initial clusters before or within step (2) that ensures the dispersion of the initial clusters.

We take Fig. 1 as an example:

If $k = 2$, the initial clusters are $S_1 = B_2 = \{d, e\}$, $S_2 = P_3 = \{g, h\}$;

If $k = 3$, the initial clusters may be $S_1 = B_2 = \{d, e\}$, $S_2 = P_3 = \{g, h\}$, $S_3 = \{x\}(x \in \{a, b, i, j\})$.

2.3.2 Constraint Function

As an intrinsic property of abstraction is information loss, an abstract model contains fewer ordering constraints than its detailed counterpart. In the attempt to satisfy the requirement of an order-preserving model transformation as well as business semantics, we furthermore consider the effect on the process control flow when assigning an activity to a cluster (candidate subprocess). Our approach designs a constraint function to guide the classifying process, which consists of two parts: *business semantics distance* and *control flow ordering conflict*.

For the first part, we introduce the virtual document (Qu et al. 2006), as Weidlich et al. (2010) has done, to represent activities and compute the semantic distance between two activities or between an activity and an activity set

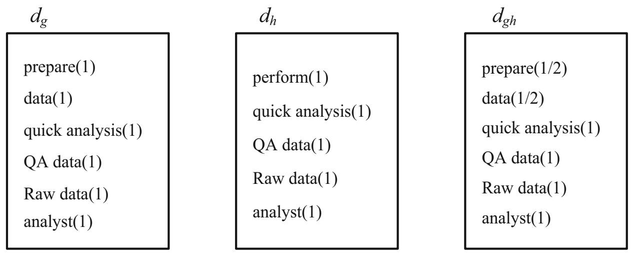

Fig. 3 The virtual documents of g, h and the subprocess composed of them

(cluster). A virtual document of a node consists of the words of all textual information that is related to that node (Weidlich et al. 2010). In our settings, the virtual document of an activity includes not only the terms of activity property labels but also the terms of all textual descriptions for this activity. Specifically, a virtual document for an activity consists of the terms that are derived from the activity label and, if this information is available, the labels of the roles that are authorized to perform the activity, the assigned input and output data, and a textual description of the activity (Weidlich et al. 2010). For a group of activities, the virtual document is derived by joining the documents of the respective nodes. The creation of virtual documents includes a normalization of terms, the filtering of stopwords, and the term stemming (Porter 1980). Given two virtual documents, their similarity can be calculated based on their distance in a vector space, in which the dimensions are the terms that appear in the documents and the values for the dimensions are computed using term frequency (Euzenat and Shvaiko 2007).

For example, given two virtual documents d_1 and d_2 represented by their term vectors $\overrightarrow{v_{d_1}}$ and $\overrightarrow{v_{d_2}}$ respectively, the similarity is defined by the cosine of the angle between the two vectors, i.e., $sim(d_1, d_2) = \cos(\overrightarrow{v_{d_1}}, \overrightarrow{v_{d_2}}) = \overrightarrow{v_{d_1}} \cdot \overrightarrow{v_{d_2}} / \left|\overrightarrow{v_{d_1}}\right|\left|\overrightarrow{v_{d_2}}\right|$. Then, the distance between two virtual documents is $dist(d_1, d_2) = 1 - sim(d_1, d_2)$. Take the activities: "*g: Prepare data for quick analysis*" and "*h: Perform quick analysis*" of Fig. 1 as an example; the virtual documents of these two activities, d_g and d_h, and the virtual document of the subprocess composed of them, d_{gh}, are respectively shown in Fig. 3. Then, the distance between d_g and d_h is $dist\left(d_g, d_h\right) = 1 - sim\left(d_g, d_h\right) = 0.27$.

Let $A = \{a_1, \ldots, a_n\}$ be the set of activities of a business process model PM and $D = \{d_1, \ldots, d_n\}$ be the corresponding virtual document set. Let $\{\mu_1, \ldots, \mu_k\}$ represent the k partition centers of the clusters $\{S_1, \ldots, S_k\}$ initialized

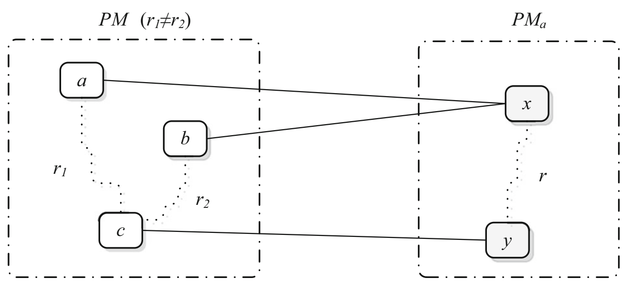

Fig. 4 An example for control flow order inconsistency

in Sect. 2.3.1. For each $a \in A$, when assigning it to a cluster S_i, we consider not only the semantic similarity (or distance) between a and μ_i, but also the possible conflicts of the control flow order with a joining S_i (the second part of the constraint function). Thus we combine the semantic similarity and the control flow order to constrain which cluster the activity should belong to. That is, when assigning the activity a to a certain cluster it may belong to, we select the S_i that minimizes the following objective function:

$$objective(S_i, a) = w_1 dist(d, \mu_i) + w_2 conflicts^*(S_i \cup \{a\}) \tag{1}$$

where d is the virtual document of a; $dist(d, \mu_i)$ represents the distance between a and the center of cluster S_i computed in terms of above virtual documents measure; $conflicts^*(S_i \cup \{a\})$ shows the possible control flow order conflict resulting from assigning a to S_i.

Each activity in the abstract model is mapped to a group of detailed activities in the original model. The control flow relation between two abstract activities may lead to order inconsistency of the corresponding detailed activities in the original model. How to deliver the control flow relations of the abstract activities is beyond the scope of this paper (see Smirnov et al. 2010a, b, c). For example, suppose a, b and c are the activities of the original model PM, a and b are respectively mapped to the abstract activity x, and c is mapped to the abstract activity y, where x and y are the activities of PM's abstract counterpart, PM$_a$. If in the abstract model the relation between x and y is r, then the relations between a and c, b and c are accordingly r. But if you suppose that in the original model the relation between a and c is r_1 and the relation between b and c is r_2, if $r_1 \neq r_2$, then this abstract model obviously results in an order inconsistency with the original model. Thus, to aggregate a and b, one of the key factors is whether or not the control flow relation between a and the other activities is consistent with that between b and the same activities, see Fig. 4.

In line with the behavioral profile, there are four order relations: $R = \{\rightsquigarrow_{PM}, \rightsquigarrow_{PM}^{-1}, +_{PM}, ||_{PM}\}$, i.e., for the above example, $r_1, r_2 \in R$. There are six different kinds of combinations of r_1 and r_2 for condition $r_1 \neq r_2$, e.g., $\rightsquigarrow_{PM} \neq +_{PM}$, and we assign a weight value to each of them to represent the *tolerance* for this inconsistency, where "1" shows no aggregation for this kind of inconsistency while "0" denotes ignorance for this kind of inconsistency.

To depict this clearly, we use a matrix W to represent the *tolerance* of all the six conflict combinations. The values of W can be prespecified according to the user's abstraction goal. Let $PM = (A, G, F, t, s, e)$ be a process model and $PM_a = (A_a, G_a, F_a, t_a, s_a, e_a)$ its abstract counterpart and BP the behavioral profile of PM. For the activities $a, b, c \in A$, suppose $\exists z \in A_a$, such that $a, b \in aggregate(z)$, $c \notin aggregate(z)$, $BP(a, c) = r_i$ and $BP(b, c) = r_j$ ($r_i, r_j \in \{\rightsquigarrow_{PM}, \rightsquigarrow_{PM}^{-1}, +_{PM}, ||_{PM}\}$), then the value of the r_i row and the r_j column, $W(r_i, r_j)$, shows the possible conflict weight value resulting from aggregating a and b into z.

For instance, we use a matrix W with strict conflict weight values in this paper and the values are provided as Eq. (2).

$$W(r_i, r_j) = \begin{cases} 0 & if\ r_i = r_j \\ 1 & otherwise \end{cases} \tag{2}$$

The corresponding matrix W is shown as follows:

$$W = \begin{array}{cccc|c} \rightsquigarrow_{PM} & \rightsquigarrow_{PM}^{-1} & +_{PM} & ||_{PM} & \\ 0 & 1 & 1 & 1 & \rightsquigarrow_{PM} \\ & 0 & 1 & 1 & \rightsquigarrow_{PM}^{-1} \\ & & 0 & 1 & +_{PM} \\ & & & 0 & ||_{PM} \end{array}$$

Users can loosen the conflict weight values according to various abstraction objectives. For example, in terms of the ratio of adding or deleting the ordering relations, we can define $W(\rightsquigarrow_{PM}, +_{PM}) = 0.5$, $W(\rightsquigarrow_{PM}, ||_{PM}) = 1/3$, $W(||_{PM}, +_{PM}) = 0.75$, and so on.

Let $S \subset A$ be a subset of A, for each activity $a_k \in A \backslash S$, the conflict value of S (as an abstract activity, a cluster or a subprocess) and a_k is computed as Eq. (3).

$$conflicts(S, a_k) = \frac{1}{|S|(|S|-1)} \sum_{\substack{a_i, a_j \in S \\ 1 \leq i \leq j \leq |S|}} W(BP(a_i, a_k), BP(a_j, a_k)) \tag{3}$$

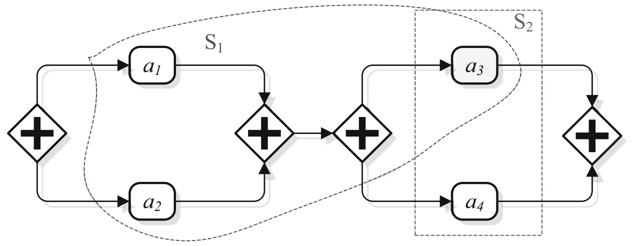

Fig. 5 A simple process PM to show the ordering conflict of activity aggregation

Table 1 The behavioral profile of PM in Fig. 5

	a_1	a_2	a_3	a_4
a_1	$+_{PM}$	$+_{PM}$	$\rightsquigarrow_{PM}$	$\rightsquigarrow_{PM}$
a_2		$+_{PM}$	$\rightsquigarrow_{PM}$	$\rightsquigarrow_{PM}$
a_3			$+_{PM}$	$+_{PM}$
a_4				$+_{PM}$

where $|S|$ denotes the number of the activities in the set S.

The control flow conflict value of S is represented by Eq. (4).

$$conflicts^*(S) = \frac{1}{|A \setminus S|} \sum_{a_k \in A \setminus S} conflicts(S, a_k) \tag{4}$$

Consider the process model *PM* of Fig. 5. The behavior profile is listed in Table 1.

When we select the cluster a_3 may belongs to, we can compute the structural conflict values of aggregating a_3 into S_1 or S_2 according to Expression (3) and (4), i.e., $conflicts^*(S_1) = 1/3$, $conflicts^*(S_2) = 0$. So if a_3 is semantically close to S_1, this value can play a regulation role for a_3 to choose a relatively reasonable cluster from the perspective of both business semantics and control flow order.

If the abstraction is realized by a human, the modeling habits of the designer are reflected in the abstraction operation as well. Hence, the values of w_1 and w_2 $(0 \le w_1, w_2 \le 1)$ may imply a designer's abstraction emphasis: if $w_1 = 1$ and $w_2 = 0$, the classification is based solely on activity business semantics; if $w_1 = 0$ and $w_2 = 1$, the classification only considers preserving the control flow order. We foresee two ways to obtain the values of w_1 and w_2: In the first way, the user explicitly specifies the values according to their emphasis; the second way implies that values are mined from a process model collection enriched with subprocess relations. We will now describe an approach in which the values of w_1 and w_2 can be discovered from such a process model collection.

Activities of a process model collection are aggregated into abstract activities, i.e., subprocess placeholders, by the model designer. The exact criteria are unknown. Yet, for each activity and each subprocess we can observe the outcome: Either the activity belongs to the subprocess or not. For a process model collection, we use the function *belong* to formalize this observation:

$$belong(a, S) = \begin{cases} 0, & \textit{if } a \in S; \\ 1, & \textit{otherwise.} \end{cases} \tag{5}$$

To mine the values of w_1 and w_2, we select them in such a way that the behavior of the function *objective* approximates the behavior of *belong*. The discovery of the values of w_1 and w_2 is realized by means of linear regression. In our setting, the values *objective* are considered independent variables and the value of the function *belong* the dependent variable. w_1 and w_2 are the regression coefficients.

2.3.3 *Constrained Clustering Algorithm for Activity Aggregation*

Based on the seeded-KMeans algorithm of Basu et al. (2002), we use the initial clusters generalized in Sect. 2.3.1 and the new objective function *objective* of Eq. (1) as the input parameters to provide a constrained clustering algorithm for BPMA.

Algorithm: Constrained-clustering-for-BPMA
Input: Set of the virtual documents, $D = \{d_1, \ldots, d_n\}$, of the corresponding activities, $A = \{a_1, \ldots, a_n\}$, included in the business process model in question; number of clusters k; set $S = \cup_{l=1}^{k} S_l$ of initial clusters (seeds); the behavior profile $BP(n \times n)$; weight matrix W; values of w_1 and w_2
Output: Disjoint k partitioning $\{C_l\}_{l=1}^{k}$ of A such that the objective function is minimized
Method:

```
1. initialize: $\mu_i^{(0)} \leftarrow \frac{1}{|S_i|}\sum_{d \in S_i} d$, $C_i^{(0)} \leftarrow S_i$, for i = 1, …, $k$; $t \leftarrow 0$
2. Repeat until convergence
  2.1 assign_cluster: Assign each activity $a$ to the cluster h*(i.e. set $C_{h^*}^{(t+1)}$),
      for $h^* = h|min\,(objective(C_h, a))$
  2.2 extimate_means: $\mu_h^{(t+1)} \leftarrow \frac{1}{|C_h^{(t+1)}|}\sum_{d \in C_h^{(t+1)}} d$
  2.3 $t \leftarrow (t+1)$
3. Output all $k$ partitions $\{C_l\}_{l=1}^{k}$
```

As the function *objective* involves not just the initial cluster centers but also the control flow relations of the activities in the initial clusters, the initial k partitioning $\{C_l\}_{l=1}^{k}$ are assigned with the set S.

We take the process model of Fig. 1 as an example to compare the proposed approach to the unsupervised clustering used in (Smirnov et al. 2011) and to analyze the advantages and limitations of the proposed approach. For brevity, we use the letters in the simplified process model of Fig. 2a to represent the corresponding activities of Fig. 1, e.g., a: Receive forecast request, b: Collect data, etc. We take the subprocess candidates shown in Fig. 1 as one of the reasonable partitions according to human designers, i.e., $k = 4$, $C_1 = \{a, b\}$, $C_2 = \{c, d, e, f\}$, $C_3 = \{g, h\}$, $C_4 = \{i, j\}$.

(1) *The effect of the initial clusters on the clustering results*

Instead of randomly generating initial cluster centers, the developed approach makes use of the Refined Process Structure Tree for decomposing the original process model and for deriving initial clusters.

For example, we respectively run the algorithm of Sect. 2.3.3 with the randomly generated initial clusters and with the ones obtained by our method of Sect. 2.3.1.

With our method, we may obtain the initial clusters: $S_1 = \{a\}$ (or $\{b\}$), $S_2 = \{d, e\}$, $S_3 = \{g, h\}$, $S_4 = \{i\}$ (or $\{j\}$). The algorithm will converge with the clusters $C_1 = \{a, b\}$, $C_2 = \{c, d, e, f\}$, $C_3 = \{g, h\}$, $C_4 = \{i, j\}$, which are consistent with the reasonable subprocess shown in Fig. 1. However, if we randomly generate the initial clusters, for example, $S_1 = \{a\}$, $S_2 = \{b\}$, $S_3 = \{c\}$, $S_4 = \{f\}$, the algorithm will output the clusters $C_1 = \{a\}$, $C_2 = \{b\}$, $C_3 = \{c, d, e, g, h\}$, $C_4 = \{f, i, j\}$ which obviously possess less reasonable business semantics and more control flow conflicts than the results derived by our approach, for instance, f integrating with i and j results in not preserving the order with g and h. Actually, we conducted twenty experiments parameterized with randomly generated initial cluster centers and only three of them output the same partitions as the subprocesses of Fig. 1 while the rest did not meet the order-preserving requirement.

(2) *The advantage of the distance measurement combining business semantics and control flow consistency*

We can see from above that the results of clustering are closely related to the initial cluster centers. However, in the proposed algorithm, the clustering is guided not only by the business semantics of activities but also by the consideration of keeping control flow order as far as possible. Even for the "bad" initial clusters, it can therefore still result in relatively reasonable clusters compared to the method (e.g., Smirnov et al. 2011) using similarity measurement solely based on semantics.

For example, with the initial clusters $S_1 = \{a\}$, $S_2 = \{b\}$, $S_3 = \{c\}$, $S_4 = \{f\}$, if we use the semantics based similarity measure such as the first part of the objective function (1) of this paper, the algorithm will converge to the clusters $C_1 = \{a, i, j\}$, $C_2 = \{b\}$, $C_3 = \{c, d, e, g, h\}$, $C_4 = \{f\}$; if we use the objective function (1) as similarity measure, then the algorithm may obtain the clusters $C_1 = \{a\}$, $C_2 = \{b\}$, $C_3 = \{c, d, e, g, h\}$, $C_4 = \{f, i, j\}$.

Though neither of the above clustering results can directly deliver a reasonable order-preserving abstract model, obviously the latter one is a better guide to generate another possible partition: $C_1 = \{a\}$, $C_2 = \{b\}$, $C_3 = \{c, d, e, g, h, f\}$, $C_4 = \{i, j\}$. Under the assumption of given desired clusters, we can also compare the two methods quantitatively, see the indexes in Sect. 3.

Of course, if we parameterize with the initial clusters obtained by our method, i.e., $S_1 = \{a\}$ (or $\{b\}$), $S_2 = \{d, e\}$, $S_3 = \{g, h\}$, $S_4 = \{i\}$ (or $\{j\}$), both similarity measure will make the algorithm converge to the clusters $C_1 = \{a, b\}$, $C_2 = \{c, d, e, f\}$, $C_3 = \{g, h\}$, $C_4 = \{i, j\}$ which is consistent with Fig. 1. But with the semantics based similarity measure, after the first loop of clustering, the distances between activity b and the four cluster centers are respectively 0.75, 1, 0.75, and 0.75. If we do not assign b to S_1 but to S_3 or S_4, it will not generate the above results. Yet with our proposed objective function (1), after the first loop of clustering, the distances between activity b and the four cluster centers are respectively 0.38, 0.52, 0.40, and 0.42, the activity b is without doubt assigned to S_1 and the algorithm converges.

(3) *Errors of clustering*

The activity clustering in this paper signifies a hard partition by which each activity has to be assigned as belonging to exactly one cluster, and when classifying an activity, the closest cluster center is selected. But in the BPMA use case of "quick view", if the abstraction is realized by a human (such as the process model collection already enriched with subprocess relations that we use for empirical analysis in Sect. 4), we find there are usually some activities belonging to no subprocesses or not assigned to the closest cluster center. We divide these kinds of activities into the following two cases according to the process fragment in Fig. 6:

1. If a is semantically closest to S_2, the k-means clustering will classify a to S_2. But due to the order-

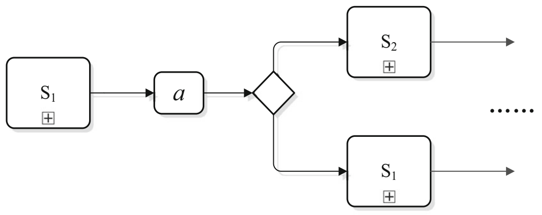

Fig. 6 An exemplified process fragment

preserving requirement, it may be classified to S_1 manually.

2. If a is semantically closest to S_1, it must be classified to S_1 based on the k-means clustering. But as it's still not close enough according to some threshold that human designers predefined, it at last belongs to no subprocesses.

We classify activities by using the distance measurement combining control flow consistency and business semantics, as in the objective function (1), but behavioral profiles are known to be problematic in the context of cycles (particularly larger ones). For instance, if we put the process fragment of Fig. 5 into a loop, then all relations between the activities become "+", so in this situation the second part of function (1) will not work anymore. The main reason for this problem is the fact that we have not considered an evaluation for the resulting abstraction model so that the hard partition cannot guarantee correctly classifying activity a of Fig. 6. Thus we can conclude that it is not sufficient to classify an activity solely based on the closest-center rule. A predefined threshold is a good constraint when deciding if an activity should be assigned to some cluster, but it is not easy to be sufficiently determined. We are considering to introduce fuzzy clustering technique in future work. The novel method will compute a fuzzy matrix regarding all activities and the cluster centers, in which each activity has at least one nonzero value for some cluster center. Based on this matrix, we can determine all *special activities* such as a of Fig. 6 and their belonging states. We enumerate all possible resulting abstraction models according to the state combination of the *special activities* and design a new index evaluating the abstraction model to identify the final clusters of these activities.

3 Empirical Analysis

In order to learn how well the proposed method approximates the abstraction results of human modelers, we performed an empirical validation of the approach by conducting an experiment with a real world business process model collection. This section provides a detailed discussion of the validation and explains in detail the experiment design and the validation results.

3.1 Validation Setup

3.1.1 Choosing the Set of Business Process Models

As a research object we chose a set of business process models from a large automobile production enterprise (China's largest automobile manufacturing organization) and its partner logistics company. This research was supported by the laboratory we work in and we have cooperated with these two companies for years. The models they provide possess normal representation and high quality, many with large number of nodes already include human-designed subprocesses. We chose 50 elaborate models enriched with the normal and relatively complete description of activity labels and activity attributes labels, of which 40 models were composed of human-designed subprocesses. To represent the activities as vector spaces by using the words of the related labels, we furthermore renormalized the terms and reached an agreement with the employees involved in our research. In addition, for achieving as much information as possible, we also considered the control flow of the processes and put the extracted label words of the adjacent activities into the virtual documents. The label words were transformed into variable names according to their meanings when they were composed of single numbers.

The processes of this automobile enterprise are very complex and there are often many subprocesses with different domains included in one process, like the subprocess of Logistics Transportation Management included in the process of Assembly Line. So intuitively we chose the process models: (1) whose sizes were moderate, (2) which included as few cross-domain subprocesses as possible, (3) which as far as possible included no more than two hierarchies of subprocesses. Table 2 outlines the relevant properties of the process models, of which the models M_1–M_{40} are enriched with human-designed subprocesses and M_{41}–M_{50} are flattened models.

3.1.2 Mining the Constraint Function

To formally validate how well the designed activity aggregation approximates the behavior of modelers clustering a set of activities into the same subprocess, we selected the following approach adopted in Smirnov et al. (2011). For each pair of an activity and a process hierarchy, we evaluated two values in the process model collection: *belong* and *objective*. Here, *belong* describes the human abstraction style, which indicates whether a certain activity is decided to be placed in the subprocess or not. The value

Table 2 The relevant properties of the process models M_1–M_{50}

	M_1–M_{40}			M_{41}–M_{50}
	Activities	Subprocesses	Activities of subprocesses	Activities
Average	94.10	7.97	7.52	71.00
Maximum	127.00	20.00	10.50	101.40
Minimum	59.00	3.00	4.20	57.00

of *objective* represents the distance between the activity and the subprocess in accordance with our approach considering both semantics and structure. To discover if the two approaches yield similar results, we studied the correlation between the two variables. A strong correlation of two variables implies that *objective* is a good constraint function in the clustering algorithm. Given the nature of the observed variables, we employed Spearman's rank correlation coefficient.

In the following, we firstly investigate the human abstraction style in the model collection as a whole. Then, we apply the K-fold cross validation process of Smirnov et al. (2011) to verify the results. We choose 30 models (M_1–M_{30}) with human-designed subprocesses and as in Smirnov et al. (2011), we also partition the model sample into four subsamples, i.e., K = 4 and perform four tests. In each test, the partition is random and three subsamples are used to discover the values of w_1 and w_2, while the fourth subsample is used to evaluate the correlation values between *belong* and *objective*.

3.1.3 Evaluating the Constrained Clustering Algorithm for BPMA

We applied the constrained clustering algorithm (*Constrained_Clustering_for_BPMA*) proposed in Sect. 2.3 and the unsupervised *k*-means clustering process (called *K-Means_for_BPMA*) to BPMA. The latter approach, similar to Smirnov et al. (2011), automatically obtained a subprocess decomposition of the flattened process model by computing the distance between an activity and a cluster only according to business semantics ("*dist*" of this paper) and initializing the cluster centers with the randomly chosen activities. We compared the abstraction results produced by these two methods. The validation of the algorithms included two parts: the first part transformed the process models M_{31}–M_{40} enriched with manually designed subprocesses into the corresponding flattened models, respectively used *Constrained_Clustering_for_BPMA* and *K-Means_for_BPMA* to generate clusters (subprocesses), and then compared the proximity degree to human designed subprocesses; the second part ran *Constrained_Clustering_for_BPMA* and *K-Means_for_BPMA* for models M_{41}–M_{50} and handed out the results to the employees who were involved in this research to evaluate and analyze.

We introduced metrics partially taken from Reijers et al. (2010) to compare the characteristics for the decomposition done by humans and the decomposition that were done automatically. The metrics was described as following:

- *Subprocesses*: total number of subprocesses in the model.
- *Avg. activities per subprocess*: average number of the activities in each subprocess.
- *Max. activities each subprocess*: maximum number of the activities in the subprocess.
- *Min. activities per subprocess*: minimum number of the activities in the subprocess.
- *Precision*: the number of subprocesses that are both automatically found and existing according to humans, divided by the number of subprocesses that are found.
- *Recall*: the number of found and existing subprocesses divided by the number of existing subprocesses.
- *Overshoot*: the fraction of found activities that does not belong in a subprocess.
- *Undershoot*: the fraction of activities that do belong but that are not found.

The precision and recall were defined in terms of the number of matched activities that constituted the subprocesses, rather than in terms of the number of (exactly) matched subprocesses. According to (Reijers et al. 2010), the metrics was computed as follows.

Let N be the set of all activities in a process (including its subprocesses), $P_M \subseteq PN$ be the set of all subprocesses that were determined manually by humans and $P_A \subseteq PN$ be the set of all subprocesses that were determined automatically, $P_M \in P_M$ be a subprocess that was determined manually by humans and $P_A \in P_A$ be a subprocess that was determined automatically. The overlap between P_A and P_M was:

$$\text{Overlap} = \frac{|P_A \cap P_M|}{\max(|P_A|, |P_M|)}$$

P_A was the most relevant match for P_M if its overlap with P_M was greater than 0 and there was no other automatically determined subprocess $P'_A \in P_A$ with a higher overlap than P_A. Let the function match: $P_M \rightarrow PN$

returned the most relevant match for each manually determined subprocess, or the empty set if no such match existed.

Precision and recall were defined as follows.

$$\text{Precision} = \frac{\sum_{P_M \in P_M} |P_M \cap \text{match}(P_M)|}{\sum_{P_A \in P_A} |P_A|},$$
$$\text{Recall} = \frac{\sum_{P_M \in P_M} |P_M \cap \text{match}(P_M)|}{\sum_{P_M \in P_M} |P_M|}.$$

The F Score was the harmonic mean of the precision and the recall: $\text{F} = 2 \times \text{precision} \times \text{recall}/(\text{precision} + \text{recall}.)$

Overshoot and undershoot were defined as follows.

$$\text{Overshoot} = \frac{\sum_{P_M \in P_M} |\text{match}(P_M) - P_M|}{\sum_{P_A \in P_A} |P_A|},$$
$$\text{Undershoot} = \frac{\sum_{P_M \in P_M} |P_M - \text{match}(P_M)|}{\sum_{P_M \in P_M} |P_M|.}$$

In the second part of the validation, we ran *Constrained_Clustering_for_BPMA* and *K-Means_for_BPMA* for models M_{41}–M_{50}. And then the results wer handed out to ten employees who were involved in our research. The employees analyzed and evaluated each of the generated abstract models based on their experiences. Each employee independently added activities or deleted the activities included in the generated subprocesses to revise them to the final subprocesses conformed to their own experiences. According to the validation method, we derived ten different values of each metric for the models M_{41}–M_{50}. We took the average value as the evaluation results of each model.

For brevity, we only recorded the number, n_{add}, of the activities that the employees added (the fraction of activities that belonged to the subprocess but were not found in the generated subprocesses) and the number, n_{del}, of the activities that the employees delete (the fraction of found activities that did not belong to the subprocess). Then the value of $|P_A| + n_{\text{add}} - n_{\text{del}}$ represented the number of activities of the subprocesss after the employees revised. If $|P_A + n_{\text{add}} - n_{\text{del}} = 0|$, then the process fragment P_A was not a meaningful subprocess, i.e., did not match any human designed subprocess.

The metrics could be deducted from the ones in the first part of the validation and computed as following, where |Group| denoted the number of the employees in the corresponding group.

$$\text{Overshoot*} = \frac{\sum_{\text{person} \in \text{Group}} \frac{\sum_{P_A \in P_A} \text{n}_{\text{del}}}{\sum_{P_A \in P_A} |P_A|}}{|\text{Group}|},$$
$$\text{Undershoot*} = \left(\sum_{\text{person} \in \text{Group}} \frac{\sum_{P_A \in P_A} \text{n}_{\text{add}}}{\sum_{P_A \in P_A} (|P_A| + \text{n}_{\text{add}} - \text{n}_{\text{del}})} \right) \Big/ |\text{Group}|,$$

$$\text{Precision*} = \frac{\sum_{\text{person} \in \text{Group}} \frac{\sum_{P_A \in P_A} (|P_A| - \text{n}_{\text{del}})}{\sum_{P_A \in P_A} |P_A|}}{|\text{Group}|},$$
$$\text{Recall*} = \left(\sum_{\text{person} \in \text{Group}} \frac{\sum_{P_A \in P_A} (|P_A| - \text{n}_{\text{del}})}{\sum_{P_A \in P_A} (|P_A| + \text{n}_{\text{add}} - \text{n}_{\text{del}})} \right) \Big/ |\text{Group}|,$$

$$\text{F}^* = \left(\sum_{\text{person} \in \text{Group}} \frac{2 \times \text{precision}^* \times \text{recall}^*}{\text{precision}^* + \text{recall}^*} \right) \Big/ |\text{Group}|.$$

3.2 Results and Analysis

Table 3 outlined the validation's results of mining the values of w_1 and w_2. The columns in the table corresponded to the function *objective*. The values of w_1 and w_2 used in *objective* were obtained using linear regression as described in the previous section. The rows of Table 3 corresponded to the experiments. Rows 1–4 described the results of 4 tests along the K-fold cross validation we explained earlier, while the last row provided the average correlations observed in the 4 separate tests. The correlation values that were presented in Table 3 were all significant when using a confidence level of 99%, i.e., all $p\psi$ values are lower than 0.01. Overall, the presented correlation values ranged around 0.7 except for that of the first test which was a little lower (0.55). This level was generally considered to indicate a strong correlation (Smirnov et al. 2011), particularly in situations where human decision making was involved. Therefore, we could speak of a strong relation between the *belong* and *objective* measures.

We mined the values of w_1 and w_2 from all models of M_1–M_{30} four times and used the average values as the parameters of Expression (1) to run *Constrained_Clustering_for_BPMA* and *K-Means_for_BPMA* and compared the abstraction results produced by these two methods.

Table 4 showed the validation results for the first part of the experiments on the process models M_{31}–M_{40} which were initially enriched with manually designed subprocesses and were transformed into the corresponding flattened models before the test. For brevity, we only give the average values of the metrics introduced in the previous section for these 10 models.

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

Experiment	$\rho(\textit{belong}, \textit{objective})$
Test_1	0.55
Test_2	0.70
Test_3	0.77
Test_4	0.68
$\text{Average}_{1–4}$	0.68

Table 4 The average values of the metrics for models M_{31}–M_{40}

Metric	*Constrained_Clustering_for_BPMA*	*K-Means_for_BPMA*	Original
Subprocesses	8.4	8.4	8.4
Avg activities per subprocess	12.57	12.59	8.31
Max activities each subprocess	24.6	34.8	15.8
Min activities per subprocess	4.7	2.5	4.1
Precision	0.53	0.32	–
Recall	0.59	0.35	–
F	0.56	0.33	–
Overshoot	0.38	0.59	–
Undershoot	0.41	0.65	–

Table 5 The average values of metrics for models M_{41}–M_{50}

Metric	*Constrained_Clustering_for_BPMA*	*K-Means_for_BPMA*	Revised
Subprocesses	9.07	9.07	6.7
Avg activities per subprocess	8.8	8.1	9
Max activities each subprocess	16.5	27	10.2
Min activities per subprocess	2.7	1	3
Precision*	0.76	0.31	–
Recall*	0.76	0.44	–
F*	0.76	0.36	–
Overshoot*	0.39	0.4	–
Undershoot*	0.24	0.35	–

Table 5 shows the validation results for the second part of the experiments on the flattened process models M_{41}–M_{50}. The number, k, of subprocesses is predetermined by modelers according to their experience.

The F-Score is the most important metric, because it provides an indication of how well a subprocess division approximates the original manual subprocess (Reijers et al. 2010). We can see from Tables 4 and 5 that the algorithm *Constrained_Clustering_for_BPMA* can be used better to approximate manual division into subprocesses than *K-Means_for_BPMA can.*

For the second part of experiment, the involved employees added or deleted activities only depending on the single cluster without considering the control flow of the whole abstract process model. So we found many revised subprocesses reused the same activities or even other subprocesses. But the F-Score of 0.76 indicates that the automatic abstraction results closely approximate the manual subprocesses.

As we used the constraint function combining semantics and structure to guide the clustering process, the maximum number of activities in each automatically generated subprocess was greatly reduced and close to the maximum number of activities in the human designed subprocess. This indicates a relatively effective control for assigning activities to a cluster (subprocess).

However, we also found that the values of Overshoot and Undershoot were relatively high in both parts of the experiment. Both *Constrained_Clustering_for_BPMA* and *K-Means_for_BPMA* are hard clustering methods which means each activity should belong to one exact subprocess. But in practical cases, we found activities not belonging to any manual subprocess. Figure 7 shows the total number of activities in models M_{31}–M_{40} and the number of those activities contained in the manual subprocesses. In the average case, *10* percent of the activities did not belong to any manual subprocess. But these activities were still clustered into the automatically generated subprocesses which partly contributed to the high values of Overshoot and Undershoot.

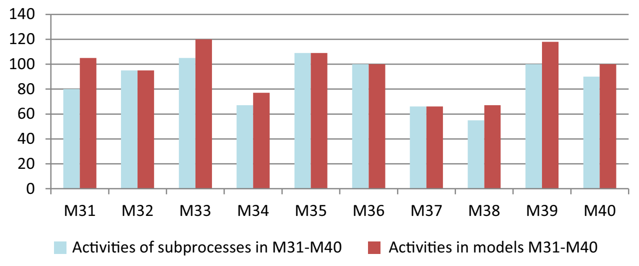

Fig. 7 Distribution of the activities in M_{31}–M_{40}

Another reason for high Overshoot and Undershoot is that, using a human designer's criterion, some activities are assigned to one subprocess S_1 even though they are semantically or structurally closer to another subprocess S_2, i.e., the distance (computed by our proposed function *objective* or other distance functions) between these activities and the subprocess they are assigned to is greater than the distance between them and the subprocess they are not assigned to. In this case, it seems insufficient to classify activities only according to the similarity of the activity and the subprocess.

4 Related Work

The topic of business process model abstraction can be related to several research streams. This paper mainly focuses on the discipline of business process management which is concerned with using methods, techniques, and software to design, enact, control, and analyze operational processes.

A large body of knowledge relates to the process model analysis based on model transformations. An example of such model transformation methods is based on structural patterns. It is a widely made observation that process models exhibit recurrent structures (van der Aalst et al. 2003; Gschwind et al. 2008; Lau et al. 2009; Smirnov et al. 2010a). The consideration of such recurrent structures facilitates several formal model analysis methods, e.g., Fahland et al. (2011) and Mendling et al. (2008) argue how recurrent structures speed up the soundness checking. The topology of the recurrent structures is described by patterns and a transformation method is specified for each pattern. Structural patterns can be used to realize process model abstraction, i.e., patterns along with the associated transformations are natural candidates for the implementation of aggregation. Smirnov (2012) defines the combination of the structural pattern and its transformation specification as an elementary abstraction. However, the identified set of process model fragment types is definitely not complete with respect to the structure of process models observed in practice. Consequently, not every process models can be abstracted by the presented set of elementary abstractions. Against this background, various research endeavors suggest broader elementary abstraction sets. For instance, the study in Polyvyanyy et al. (2008) complements sequence, block, and loop elementary abstractions with the dead end elementary abstraction. Bobrik et al. (2007b), Dumas et al. (2010), and Liu and Shen (2003) advocate more sophisticated elementary abstractions.

But each elementary abstraction set requires an argument concerning the model class reducible with the given elementary abstractions. The need for such an argument is the main limitation of pattern-based approaches. 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 Vanhatalo et al. (2009), 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 be used for abstraction in process models (Polyvyanyy et al. 2009b).

Finally, one can distinguish model transformations that preserve process behavior properties. In Van der Aalst and Basten (1997), the authors introduce three notions of behavioral inheritance for WF-nets and study inheritance properties. The paper suggests model transformations, in which the resulting model inherits the behavior of the initial model. An approach for process model abstraction can exploit such transformations as basic operations. Kolb and Reichert (2013a) have introduced a framework for enabling order-preserving process model abstractions based on parameterized aggregation and reduction operations. In particular, these operations may be configured in different ways to either preserve the behavior of the original process model or to allow for some relaxations (i.e., order constraint violations) depending on the respective application context.

The outlined model transformations can support a solution of the general problem of process model abstraction, but they all focus on structural and behavioral aspects of models and model transformations, disregarding the semantic aspect.

Semantic aggregation of activities relates to research on semantic business process management, and process models enriched with semantic information facilitate many process analysis tasks, see (Hepp et al. 2005). Along this line of research, several authors describe how to use activity ontologies to realize activity aggregation (Casati and Shan 2002; Alves de Medeiros et al. 2008). It should be noted, however, that such approaches imply the existence of a semantic description for model elements and their relations, which is a restriction that rarely holds in real world settings. Smirnov et al. (2010b) present a semi-automated approach for activity aggregation that reduces the human effort. However, it requires the help of predefined information external to the model: a domain ontology specifying activity meronymy relations to evaluate the activity relatedness. Smirnov et al. (2011) provide an approach that exploits semantic information within a process model, beyond structural information, to decide which activities belong to one another. The approach aggregates activities solely according to the business semantics without discussing control flow, and the abstraction style is mined from one model collection of the particular domain which makes the distance measure not sufficiently general. Weidlich et al. (2010) propose an approach to identify a group of activities functionally similar to a given activity. But the matching is between two distinct models so that it is not able to be directly applied to the activities within one singular model. Reijers et al. (2010) investigate three types of criteria when deciding whether nodes should be put together into a subprocess: block-structuredness takes the canonical components of RPST (Vanhatalo et al. 2009) as candidate subprocesses; the connectedness criterion uses graph cluster analysis (Schaeffer 2007) to establish collections of nodes that are strongly connected to each other in a business process. These two criteria discover subprocesses based on structure and many generated subprocesses are too large or too small or incomplete from the perspective of business semantics. The label similarity criterion builds on the idea that nodes with more similar labels can be considered to have a higher probability of belonging to the same subprocess than nodes that have very different labels. But to only depend on the activity names is not enough to show the similarity relations between the activities.

Structure-based business process model abstraction tends to be suitable for the situation of user control where a user can determine which activities are significant and which are irrelevant. Then the abstraction operation hides the irrelevant ones by concealing them in structure patterns or decomposed components. In the context of requiring a quick view of a process model, the abstraction is completely out of user control to provide all the subprocess candidates with meaningful business semantics. In such a case, the business process model abstraction only based on structure cannot answer questions such as "how to discover the semantics related activities" or "whether or not the candidate subprocess is meaningful for business". Semantics-based business process model abstraction proposes discovering aggregated activities from the perspective of the business semantics, however, it solely considers the semantics leaving the control flow out of regard so that quite a lot of activities in one candidate subprocess are not structure-connected, resulting in the lack of order preservation in the abstract models. We proposed an approach to extend the canonical components of RPST by discovering the set of activities whose semantic description is most similar to the extended canonical component (see Nan et al. 2015). But the discovering activity is restricted to the nodes which are directly adjacent to the canonical components of RPST which makes this method more similar to structure-based abstraction. At present there is no explicit research combining business semantics and control flow to guide the activity aggregation.

Establishing an activity's granularity level is also a recurrent challenge in process mining, where logs contain records that are often very fine-granular. Process mining refers to the extraction of process models from event logs (van der Aalst et al. 2004). Real-life processes tend to be less structured than expected. As such, the process models directly mined from the logs can be overloaded with information making them hard to comprehend. Therefore, some researchers have proposed abstraction techniques to improve the mined models. In Bose and van der Aalst (2009), the authors showed that common execution patterns (e.g., tandem arrays, maximal repeats etc.) manifested in an event log can be used to create abstractions and that these abstractions are used in their two-phase approach to process discovery (Li et al. 2010) as a pre-processing of the event log enabling the discovery of hierarchical process models. But the patterns they defined are closely related to the process control flow and depend on the availability of rich process logs. In our approach the activities to be abstracted are discovered by considering not only the control flow consistency but also the business semantics implied in human's design standards and we generate them based on a detailed process model's ontology description without needing any process logs. Actually, models are rarely enriched with such detailed execution information (Smirnov et al. 2012).

In Bose et al. (2012), the authors demonstrate the discovery of hierarchical process models using a chain of plugins implemented in ProM. The (enhanced) fuzzy miner plugin (Günther and van der Aalst 2007) is applied on the transformed log. Günther and van der Aalst (2006, 2007) propose activity aggregation mechanisms based on clustering algorithms. The mechanisms extensively use information present in process logs, i.e., timestamps of activity starts and stops, activity frequencies, and transition probabilities, which however are less common for process models. Thus, in contrast to the activity aggregation approach proposed in our paper, process mining considers other activity property types for clustering and utilizes different clustering algorithms.

5 Conclusions and Future Work

Business process model abstraction has been addressed in a number of research endeavors, but we propose a novel approach to this area. Our main contribution is a method based on constrained *k*-means clustering analysis to discover sets of related activities taking into account both business semantics and control flow ordering, where each set corresponds to a coarse-grained activity of an abstract process model. As a second contribution, we propose an approach that mines the clustering constraint from a given process model collection, which is reusable for abstraction of new process models. 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, it builds on the assumption that the number of subprocesses k can be predetermined; however, in practice this number is difficult to identify based on modelers' experiences. The experimental validation of Table 5 and Fig. 7 shows that the number of revised subprocesses is not exactly equal to the pre-specified number of subprocesses (an estimated value according to the modelers' experiences).

Secondly, k-means clustering is a hard partition for the data set which means that each activity should belong to one subprocess. But in practice, activities exist which do not belong to any manual subprocess. And although some activities are closer to one subprocess based on our proposed objective function, they are assigned to another subprocess when manually classified.

Thirdly, we utilized as much information as possible to compute the similarity between activities, but when delivering an abstract process model, we only considered control flow and disregarded other perspectives (e.g., data objects, data flow, and resources).

These and other limitations guide our future research plans. The very next step for us is to design a proper evaluation index to assess the process abstraction results and generate the optimal number of subprocesses. A further step will be to apply and improve soft clustering techniques, such as Fuzzy C-Means (FCM) clustering, to replace k-means clustering, so that we can assign the activities to a subprocess more flexibly. Furthermore, we can explore how other perspectives (e.g., Kolb and Reichert 2013b), support control flow when abstracting a process model.

Acknowledgements This work is supported in part by NSFC under Grant Nos. 61402193, 61272208, 61133011, 60973089, 61003101, 61170092, by the Jilin Province Science and Technology Development Plan under Grant Nos. 20130522177JH, by the Jilin Provincial Department of Education "Twelfth/Thirteenth Five Year Plan" Science and Technology Development Plan under Grant Nos. 2014160, 2016105, and by the Jilin Province Education Science "Twelfth Five Year Plan" under Grant Nos. GH150285, GH16249.

References

Alves de Medeiros AK, van der Aalst WMP, Pedrinaci C (2008) Semantic process mining tools: core building blocks. In: Proceedings of the 16th European conference on information systems, Galway, pp 475–478

Bar-Hillel A, Hertz T, Shental N (2003) Learning distance functions using equivalence relations. In: Proceedings of the twentieth international conference on machine learning, pp 11–18

Basu S, Banerjee A, Mooney RJ (2002) Semi-supervised clustering by seeding. In: Proceedings of the nineteenth international conference on machine learning, pp 19–26

Basu S, Bilenko M, Mooney RJ (2004) A probabilistic framework for semi-supervised clustering. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining, pp 59–68. doi:10.1145/1014052.1014062

Bilenko M, Basu S, Mooney RJ (2004) Integrating constraints and metric learning in semi-supervised clustering. In: Proceedings of the twenty-first international conference on machine learning, pp 81–88. doi:10.1145/1015330.1015360

Bobrik R, Reichert M, Bauer T (2007a) View-based process visualization. In: International conference on business process management, Brisbane, Australia. LNCS 4714. Springer, Heidelberg, pp 88–95

Bobrik R, Reichert M, Bauer T (2007b) Parameterizable views for process visualization. Technical report TR-CTIT-07-37, Centre for Telematics and Information Technology, University of Twente, Enschede

Bose RPJC, van der Aalst WMP (2009) Abstractions in process mining: a taxonomy of patterns. In: Proceedings of the 7th international conference on business process management. LNCS 5701. Springer, Heidelberg, pp 159–175

Bose RPJC, Verbeek EHMW, van der Aalst WMP (2012) Discovering hierarchical process models using ProM. In: Nurcan S (ed) IS Olympics: information systems in a diverse world, vol 107. LNBIP, pp 33–48

Casati F, Shan M-C (2002) Semantic analysis of business process executions. Proceedings of the 8th international conference on extending database technology: advances in database technology. Springer, Heidelberg, pp 287–296

Cohn D, Caruana R, McCallum A (2009) Semi-supervised clustering with user feedback. In: Basu S Davidson I, Wagstaff K (eds) Constrained clustering: advances in algorithms, theory, and applications. Data Mining and Knowledge Discovery Series, chapter 2. CRC, Boca Raton, pp 17–31

Demiriz A, Bennett KP, Embrechts MJ (1999) Semi-supervised clustering using genetic algorithms. In: Proceedings of the artificial neural networks in engineering conference, pp 809–814

Dumas M, Luciano García-Bañuelos L, Polyvyanyy A et al (2010) Aggregate quality of service computation for composite services. In: ICSOC 2010, San Francisco, 7–10 December. LNCS 6470. Springer, Heidelberg, pp 213–227

Eshuis R, Grefen P (2008) Constructing customized process views. Data Knowl Eng 64(2):419–438

Euzenat J, Shvaiko P (2007) Ontology matching. Springer, Heidelberg

Fahland D, Favre C, Koehler J et al (2011) Analysis on demand: instantaneous soundness checking of industrial business process models. Data Knowl Eng 70(5):448–466

Francescomarino CD, Marchetto A, Tonella P (2013) Cluster-based modularization of processes recovered from web applications. J Softw Maint Evol Res Pract 25(2):113–138

Gao Y, Liu DY, Qi H (2008) Semi-supervised k-means clustering algorithm for multi-type relational data. J Softw 19(11):2814–2821 (in Chinese with English abstract)

Gaynor S, Bair E (2013) Identification of biologically relevant subtypes via preweighted sparse clustering. ArXiv e-prints 2013. arXiv:1304.3760. http://biostats.bepress.com/cgi/viewcontent.cgi?article=1032&context=uncbiostat. Accessed 14 Aug 2014

Gschwind T, Koehler J, Wong J (2008) Applying patterns during business process modeling. In: International conference on business process management. LNCS 5240. Springer, Heidelberg, pp 4–19

Günther CW, van der Aalst WMP (2006) Mining activity clusters from low-level event logs. BETA working paper series, WP 165

Günther CW, van der Aalst WMP (2007) Fuzzy mining: adaptive process simplification based on multi-perspective metrics. In:

International conference on business process management, Brisbane, LNCS 4714. Springer, Heidelberg, pp 328–343

Hastie T, Tibshirani R, Friedman JH (2009) The elements of statistical learning: data mining, inference, and prediction, 2nd edn. Springer, Heidelberg

Hepp M, Leymann F, Domingue J et al. (2005) Semantic business process management: a vision towards using semantic web services for business process management. In: IEEE international conference on e-business engineering (ICEBE'05), Beijing. IEEE Computer Society, pp 535–540

Kamvar SD, Klein D, Manning CD (2003) Spectral learning. In: Proceedings of the 18th international joint conference on artificial intelligence. Morgan Kaufmann, pp 561–566

Klein D, Kamvar SD, Manning CD (2002) From instance-level constraints to space-level constraints: making the most of prior knowledge in data clustering. In: Proceedings of the nineteenth international conference on machine learning. Morgan Kaufmann, Burlington, pp 307–314

Kolb J, Reichert M (2013a) A flexible approach for abstracting and personalizing large business process models. ACM Sigapp Appl Comput Rev 13(1):6–17

Kolb J, Reichert M (2013b) Data flow abstractions and adaptations through updatable process views. In: Proceedings of 28th ACM symposium on applied computing. ACM, pp 1447–1453

Lau JM, Iochpe C, Thom L et al (2009) Discovery and analysis of activity pattern co-occurrences in business process models. In: Proceedings of the international conference on enterprise information systems, Milan, vol Isas, pp 83–88

Li J, Bose RPJC, van der Aalst WMP (2010) Mining context-dependent and interactive business process maps using execution patterns. In: zur Muehlen M, Su J (eds) BPM 2010 Workshops, LNBIP 66. Springer, Heidelberg, pp 109–121

Liu D, Shen M (2003) Workflow modeling for virtual processes: an order- preserving process-view approach. Inf Syst 28(6):505–532

Mendling J, Verbeek H, van Dongen BF et al (2008) Detection and prediction of errors in EPCs of the SAP reference model. Data Knowl Eng 64(1):312–329

Nan W, Shanwu S, Ying L et al (2015) Business process model abstraction based on structure and semantics. ICIC Express Lett 2(9):557–563

Polyvyanyy A, Smirnov S, Weske M (2008) Reducing complexity of large EPCs. In: EPK 2008 GI-Workshop, Saarbrücken

Polyvyanyy A, Smirnov S, Weske M (2009a) On application of structural decomposition for process model abstraction. Business Process, Services Computing and Intelligent Service Management, Leipzig, pp 110–122

Polyvyanyy A, Smirnov S, Weske M (2009b) The triconnected abstraction of process models. In: International Conference on Business Process Management, Ulm, LNCS 5701. Springer, Heidelberg, pp 229–24

Polyvyanyy A, Vanhatalo J, Völzer H (2010) Simplified computation and generalization of the refined process structure tree. In: Proceedings of the WS-FM 2010, LNCS 6551. Springer, Heidelberg, pp 25–41

Porter MF (1980) An algorithm for suffix stripping. Progr 14(3):130–137

Qu Y, Hu W, Cheng G (2006) Constructing virtual documents for ontology matching. In: Proceedings of the 15th international conference on World Wide Web. ACM, New York, pp 23–31. doi:10.1145/1135777.1135786

Reijers HA, Mendling J, Dijkman RM (2010) On the usefulness of subprocesses in business process models. BPM center report BPM-10-03. http://www.BPMcenter.org. Accessed 18 Sept 2013

Ruiz C, Spiliopoulou M, Menasalvas E (2007) C-DBSCAN: density-based clustering with constraints. In: Proceedings of the Rough sets, fuzzy sets, data mining and granular computing. LNCS 4482, pp 216–223. doi:10.1007/978-3-540-72530-5_25

Salton G, Wong A, Yang CS (1975) A vector space model for automatic indexing. Commun ACM 18(11):613–620

Schaeffer S (2007) Graph clustering—survey. Comput Sci Rev 1:27–64

Schultz M, Joachims T (2003) Learning a distance metric from relative comparisons. Adv Neural Inf Process Syst 16:40–47

Sharp A, McDermott P (2008) Workflow modeling: tools for process improvement and applications development. Artech House, London

Smirnov S (2012) Business process model abstraction. Doctor Dissertation, University of Potsdam

Smirnov S, Weidlich M, Mendling J et al (2009) Action patterns in business process models. Comput Ind 63(2):115–129

Smirnov S, Weidlich M, Mendling J (2010) Business process model abstraction based on behavioral profiles. Service-Oriented Computing. LNCS 6470. Springer, Heidelberg, pp 1–16

Smirnov S, Weidlich M, Mendling J (2010a) Object-sensitive action patterns in process model repositories. In: zur Muehlen M et al. (eds) Business Process Management Workshops, vol 66. Springer, Heidelberg, pp 251–263

Smirnov S, Dijkman R, Mendling J et al. (2010b) Meronymy-based aggregation of activities in business process models. In: Conceptual Modeling – ER 2010. 29th international conference on conceptual modeling, Vancouver, Canada. LNCS 6412. Springer, Heidelberg, pp 1–14

Smirnov S, Reijers HA, Weske M (2011) A semantic approach for business process model abstraction. International Conference on Advanced Information Systems Engineering, LNCS 6741. Springer, Heidelberg, pp 497–511

Smirnov S, Reijers HA, Weske MH et al (2012) Business process model abstraction: a definition, catalog, and survey. Distrib Parallel Databases 30(1):63–99

Tang W, Xiong H, Zhong S et al. (2007) Enhancing semi-supervised clustering: A feature projection perspective. In: Proceedings of the thirteenth international conference on knowledge discovery and data mining, pp 707–716, doi: 10.1145/1281192.1281268

van der Aalst WMP, Basten T (1997) Life-cycle inheritance: a petri-net-based approach. Proceedings of the 18th international conference on application and theory of Petri Nets, LNCS 1248. Springer, Heidelberg, pp 62–81

van der Aalst WMP, ter Hofstede AHM, Kiepuszewski B et al (2003) Workflow patterns. Distrib Parallel Databases 14:5–51

van der Aalst W, Weijters A, Maruster L (2004) Workflow mining: discovering process models from event logs. IEEE Trans Knowl Data Eng 16(9):1128–1142

Vanhatalo J, Völzer H, Leymann F (2007) Faster and more focused control-flow analysis for business process models through SESE decomposition. In: ICSOC 2007, Vienna, LNCS 4749, pp 43–55

Vanhatalo J, Völzer H, Koehler J (2009) The refined process structure tree. Data Knowl Eng 68(9):793–818. doi:10.1016/j.datak.2009.02.015

Wagstaff K, Cardie C (2000) Clustering with instance-level constraints. In: ICML'00 proceedings of the seventeenth international conference on machine learning, pp 1103–1110

Wagstaff K, Cardie C, Rogers S et al. (2001) Constrained k-means clustering with background knowledge. In: Proceedings of the eighteenth international conference on machine learning, pp 577–584

Wang L, Bo LF, Jiao LC (2007) Density-sensitive semi-supervised spectral clustering (in Chinese with English abstract). J Softw 18(10):2412-2422. doi:10.1360/jos182412. http://www.jos.org.cn/1000-9825/18/2412.html. Accessed 21 Mar 2013

Weidlich M, Dijkman R, Mendling J (2010) The ICoP framework-identification of correspondences between process models. In:

Proceedings of the 22nd international conference on advanced information systems engineering, LNCS 6051, pp 483–498

Weidlich M, Mendling J, Weske M (2011) Efficient consistency measurement based on behavioural profiles of process models. IEEE Transact Softw Eng 37(3):410–429

Xing EP, Ng AY, Jordan MI et al (2003) Distance metric learning with application to clustering with side-information. Adv Neural Inf Process Syst 15:505–512

Xu QJ, Desjardins M, Wagstaf K (2005) Constrained spectral clustering under a local proximity structure assumption. In: Proceedings of the eighteenth international Florida artificial intelligence research society conference, Clearwater Beach, pp 866–867

Yin X, Chen S, Hu E et al (2010) Semi-supervised clustering with metric learning: an adaptive kernel method. Pattern Recognit 43(4):1320–1333. doi:10.1016/j.patcog.2009.11.005