Clustering Business Process Activities for Identifying Reference Model Components

Jana-Rebecca Rehse and Peter Fettke

Institute for Information Systems (IWi) at the German Center for Artificial Intelligence (DFKI GmbH) and Saarland University
Campus D3.2, Saarbrücken, Germany
{Jana-Rebecca.Rehse, Peter.Fettke}@iwi.dfki.de

Abstract. Reference models are special conceptual models that are reused for the design of other conceptual models. They confront stakeholders with the dilemma of balancing the size of a model against its reuse frequency. The larger a reference model is, the better it applies to a specific situation, but the less often these situations occur. This is particularly important when mining a reference model from large process logs, as this often produces complex and unstructured models. To address this dilemma, we present a new approach for mining reference model components by vertically dividing complex process traces and hierarchically clustering activities based on their proximity in the log. We construct a hierarchy of subprocesses, where the lower a component is placed the smaller and the more structured it is. The approach is implemented as a proof-of-concept and evaluated using the data from the 2017 BPI challenge.

Keywords: Reference Model Mining, Activity Clustering, Reference Components, Reference Modeling, Process Mining

1 Introduction

Reference models can be considered as special conceptual models that serve to be reused for the design of other conceptual models. By providing a generic template for the design of new process models in a certain industry, reference process models allow organizations to adapt and implement the respective processes in a resource-efficient way [1]. The introduction of a common terminology and the subsequent simplification of communications along with the industry-specific experience contained in a reference model yield higher-quality processes and process models, while also reducing the required time, cost, and personnel resources required for business process management [2].

When developing a model for the purpose of reuse, model designers are faced with the dilemma of balancing the scope of a model, i.e. its size, specificity, and degree of coverage against its reuse potential, i.e. the number of situations where it can be applied. The larger and the more specific a model is, the less adaptations it needs to be applied in a certain model context, but the less often

these situations occur. On the contrary, smaller reference models for subprocesses, named reference components, can be directly applied to many different situations, but do not suffice to cover the entire modeling domain.

We consider reference components as frequently appearing process model building blocks, i.e. temporally and logically isolated activity sets within a process [3]. For process designers, such building blocks strike a balance between the necessity to find a reference model for their exact use case and the disadvantages that come with modeling a process from scratch. Due to only a limited number of predefined interaction points with other process parts, they are frequently reusable and highly flexible to be combined into new process models [3]. By using pre-defined domain-specific collections of process building blocks, such as the PICTURE method for public administration modeling, process designers are able to leverage the multiple benefits of reference modeling and simultaneously adress the specific challenges of their own process domain [4].

However, constructively using reference model components for all stakeholders' advantage first requires finding the right degree of specificity versus reusability. As this decision depends on the intended domain and purpose, it cannot be universally determined, but needs to be decided individually for each use case. In order to provide process designers with useful and reliable data to support the decision-making process, this contribution presents a novel approach for mining reference model components from instance-level data. A given input log is vertically divided and the activities are hierarchically clustered based on their spatial proximity in the log, determining the groups of activities that form a reference component. The components are mined for each cluster individually, resulting in a subprocess hierarchy, where the lower a component is placed the smaller, but the more structured and frequent it is.

This article is based on ideas for subprocess identification sketched in a report submitted to the 2017 BPI challenge [5]. We describe the conceptual design in Sect. 2. The realization in the RefMod-Miner research prototype is treated in Sect. 3, along with an experimental evaluation. We report on related work in Sect. 4, before concluding the article with a discussion in Sect. 5.

2 Conceptual Design of the Approach

2.1 Illustrating Example and Outline

The objective of this paper is to provide a data-based solution to determining the appropriate degree of specificity versus reusability when designing reference model components, i.e. to overcome the dilemma that the larger and more specific a reference model is, the less situations it applies to. Figure 1 illustrates our solution by means of an exemplary company-specific invoice handling process, executed e.g. by an accounting clerk. Once an invoice is received and processed, it is checked. If no further action is required (e.g. for a pro-forma-invoice), the invoice is archived directly. If the payment amount exceeds a certain limit, the invoice gets forwarded to the superior, as the clerk is not authorized for payment. If the limit is not exceeded, the clerk pays and archives the invoice.

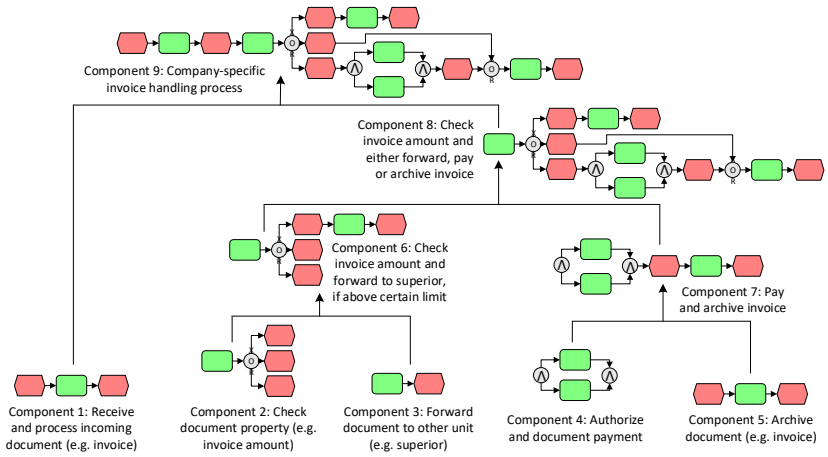

Fig. 1. Illustrating Example for Mining Reference Components

While the complete process model can be used for the design of other models, its application scope is limited to invoice handling. If, however, we divide it into its subprocesses, we see that they can be generalized to apply in other contexts. This is illustrated by the cluster structure in Fig. 1, which gradually divides the specific process into smaller parts. The smaller the parts get, the more generic they are. For example, Component 1, where the invoice is received and processed, could be part of any other invoice handling process, but could also be slightly abstracted to serve as a generic document handling subprocess.

Our approach consists of four major steps, described in detail in the following subsections. Individual activities are identified from the provided event log and clustered hierarchically based on a chosen proximity measure, resulting in a tree structure, where the higher a cluster is located, the more and less interrelated activities it contains. For each cluster, a reference component is mined, constructing a model hierarchy in analogy to the cluster structure. The higher a reference component is located, the more activities it contains and thus, the more specific it is.

2.2 Identifying Activities from Event Logs

Definition 1 (Activities, Events, and Trace [6]). *Let $\mathcal{A}$ be the activity universe and $A \subseteq \mathcal{A}$ a set of activities. An* event e^a *denotes the execution of an activity $a \in A$. A* trace $t = (e_1, ..., e_n)$ *is a finite sequence of events.*

Definition 2 (Log [6]). *A* log $L = \{(t_1)^{m_1}, ..., (t_n)^{m_n}\}$ *is a multiset of traces. The multiplicity m_i of trace t_i denotes the frequency of t_i in L. $\mathcal{A}(L) = \{a \mid \exists t \in L : e^a \in t\}$ denotes the set of activities that are contained in a log. For a set of activities $A \subset \mathcal{A}(L)$, $T_A = \{t \in L \mid \forall a \in A : e^a \in t\}$ denotes the set of traces that contain all activities in A.*

Since the objective is to cluster activities into reference model components, the first step is to extract higher-level activities from the lower-level event logs. This is non-trivial, because the events in a process log often do not correspond to the activities in process models, which are necessary to analyze them from a business perspective. The problem is recognized and described in literature, with proposed solutions making use of either data mining or machine learning techniques [7, 8] or leveraging predefined process knowledge in a supervised abstraction approach [9] to identify event patterns and automatically label the corresponding activities. As all of these approaches offer promising results, we base our approach on their capabilities to extract a meaningful set of activities from the provided event log, should that be required. Stakeholders involved in the reference component design process can provide the necessary domain knowledge.

2.3 Activity Clustering based on Spatial Proximity

For clustering the activities into a hierarchical structure, we measure the spatial proximity between activities. The idea behind this measure is that activities which often appear in close proximity to each other form a logical unit with a clear structure. For determining the similarity between two activities, we select the set of traces containing both activities at least once. For each trace, the similarity is calculated by counting the number of steps between the two activities, dividing it by the length of the trace to get a normalized value, and deducting the result from 1. If the activities are associated with multiple events within one trace, the average distance is calculated.

Definition 3 (Trace-based Spatial Proximity). *Let L be a log, $\mathcal{A}(L)$ its activity set, $a, b \in \mathcal{A}(L)$ two activities, and $T_{ab} \subseteq L$ the set of traces that contain both a and b. For a trace $t \in T_{ab}$, let $I = \{i \in \mathbb{N} | t \ni e_i = e^a\}$ and $J = \{j \in \mathbb{N} | t \ni e_j = e^b\}$ be the event index sets for activities a and b. The trace-based spatial proximity $p : \mathcal{A} \times \mathcal{A} \to [0, 1]$ between two activities is defined as*

$$p(a,b) = \begin{cases} \dfrac{\sum_{t \in T_{ab}} (1 - \dfrac{\sum_{i \in I} \sum_{j \in J} |i - j|}{|t|})}{|T_{ab}|} & \text{if } |T_{ab}| \geq 1, \\ 0 & \text{otherwise.} \end{cases}$$

The pairwise spatial proximity in form of a similarity matrix is used as input for clustering. A hierarchical-agglomerative clustering approach allows us to inspect activity clusters on different size and specificity levels [10]. The result is a strict cluster hierarchy, with the singular activities as leaves and the complete activity set as root cluster. Each internal node cluster contains the union of activities that are contained in its two child clusters. To get a precise cluster result, we do not parametrize the expected number of clusters, which increases the runtime complexity. Compared to e.g. trace clustering, where a few thousand

clustering objects are still computationally feasible [11], activity clustering typically contains not more than a hundred objects, so computation times should not become a problem. The result of a clustering is an activity hierarchy.

Definition 4 (Activity Hierarchy). *Let L be a log, $\mathcal{A}(L)$ its set of activities. An* activity hierarchy *$H_{\mathcal{A}(L)}$ is a connected, directed, acyclic graph $H = (A, E)$ (i.e. a tree), where $2^{\mathcal{A}(L)} \supset A = A_1, ..., A_n$ is a set of subsets of $\mathcal{A}(L)$ and $E \subset A \times A$ is a set of edges connecting them, such that:*

- *$(A_i, A_j) \in E \Rightarrow A_j \subseteq A_i$, i.e. a set is fully contained in its parent set,*
- *$\forall i : \bigcup_{(A_i, A_k) \in E} A_k = A_i$, i.e. a set is equal to the unification of its children.*

2.4 Mining Reference Model Components

After obtaining the cluster hierarchy, we mine a reference model component for each identified activity set. Therefore, we use our RMM-2 approach for reference model mining based on execution semantics, adapted to work with process traces instead of process models [11]. It analyzes the represented process semantics in terms of behavioral profiles and computes a reference model subsuming the specified behavior. To apply the adapted RMM-2 for successfully mining reference components, the input data has to be modified, such that the reference components contain the same activities as the associated cluster. Therefore, we compute log projections for each cluster activity set.

Definition 5 (Log Projection). *Let L be a log, $\mathcal{A}(L)$ its set of activities, $A \subseteq \mathcal{A}(L)$ a set of activities, and $T_A \subseteq L$ the set of traces that contain all activities in A. The log projection $L_A = \{t \mid \exists t' \in T_A : \forall e^a \in t' : a \in A \Rightarrow e^a \in t\}$ is the subset of T_A that contains only the activities in A.*

For an activity set A and its projected log L_A, the following steps are executed for computing a reference component:

1. Compute Behavioral Profiles: For each trace in L_A, a separate behavioral profile is computed using the trace-based activity relations.
2. Integrate Behavioral Profiles: The set of individual behavioral profiles is merged into an integrated behavioral profile, representing typical behavior from the input traces. The noise level parameter specifies the minimum threshold for a relation to be included in the integrated profile.
3. Derive Reference Component: Finally, the semantics represented in the behavioral profile are conveyed into a process model in form of an event-driven process chain (EPC).

Formally, mining the reference components assigns a reference component in form of a process model (in this case, an EPC) to each activity set contained in the activity hierarchy. In the following, we define an *process model* as a tuple $P = (N, E, type)$, according to the definition by [12, p. 92].

Definition 6 (Reference Model Hierarchy). *Let L be a log, $\mathcal{A}(L)$ its set of activities, and $H_{\mathcal{A}(L)} = (A, E)$ an activity hierarchy. The corresponding* reference model hierarchy *$RM_{H_{\mathcal{A}(L)}}$ is a function that assigns each activity set $A_i \in A$ to a reference model RM_i in form of an process model $P = (A_i, E, type)$.*

2.5 Evaluating Reference Model Components

Finally, we need to provide stakeholders with concise data to find an optimal solution for the dilemma of reusability for their use case. Reference components are supposed to contain frequently appearing model parts, so that they can be reused and applied in different contexts. They are also supposed to be internally coherent, i.e. the contained activities should exhibit some kind of correlation with one another. Applying this to our reference component hierarchy, we need to find those cluster integration steps, where combining two clusters into one does not produce a viable reference component, because the activity set is either not sufficiently frequent or not sufficiently interconnected anymore. While the former can be assessed by measuring the difference in relative frequency of the activity sets in the log, the latter can be assessed by comparing the size of the integrated reference component with the sum of the individual component sizes. Therefore, we define the following measures.

Definition 7 (Diffusion Rate and Inflation Rate). *Let L be a log, $\mathcal{A}(L)$ its set of activities, $H_{\mathcal{A}(L)} = (A, E)$ an activity hierarchy, and $RM_{H_{\mathcal{A}(L)}}$ a reference model hierarchy. Let $A_i \in A$ be a an activity set, $A_j, A_k \in A$ its children, and RM_i, RM_j, RM_k the corresponding reference models.*

The diffusion rate $\mathcal{D}_{A_i}$ defines the ratio between the number of traces that contain A_i, $T_{A_i} = T_{A_j} \cap T_{A_k}$ and the number of traces that contain its respective children, $T_{A_j} \cup T_{A_k}$.

$$\mathcal{D}_{A_i} = \frac{|T_{A_i}|}{|T_{A_j} \cup T_{A_k}|}$$

The inflation rate $\mathcal{I}_{A_i}$ defines the ratio between the size of RM_i, $|RM_i| = |A_i|$ and the added sizes of RM_j, RM_k.

$$\mathcal{I}_{A_i} = \frac{|RM_i|}{|RM_k| + |RM_j|}$$

By measuring these two relations for each reference component, we determine those points where an integration is not useful anymore, because the two components to be merged are so different from one another. They either conjointly appear in very few traces, such that the intersection of the two trace sets is much smaller than the union, resulting in low a diffusion rate, or the size of the merged component is much larger than the sum of the child components, meaning that many connector nodes are required to merge them, resulting in an inflation rate considerably higher than 1. One could argue that if many connector nodes are required for merging two components, there is a high degree of interconnection between them, requiring many interfaces. We don't want to completely rule out such a scenario, but the question arises why the two components were not clustered together in the first place, if they are so closely connected. The components' eligibility as reference components would be limited in such a case, due to their complicated structure. In general, these are only structural metrics for assessing the degree of interrelation between process model activities and may not replace the final decision by the reference model designer.

3 Proof-of-Concept and Experimental Evaluation

In order to demonstrate the capabilities of the suggested approach, it was prototypically implemented as a proof-of-concept in the RefMod-Miner research prototype, a Java-based software tool for process model analysis developed in our research group (`https://refmod-miner.dfki.de`). This implementation was used for an evaluation using the "application event log" from the 2017 BPI Challenge [13], describing a loan application process from a Dutch financial institute. We distinguish A-type events (subprocess of application handling), O-type events (offer creation), and W-type events (workflow activities). The log contains 561,671 events from 31,509 individual loan application cases.

The first step is to identify the activities. As we see from inspecting the log, the events are recorded on a lower level, but are clearly associated with a higher level activity ("concept:name"). The event itself is specified by the attribute "lifecycle:transition" and the separate ID. Tab. 1 lists the extracted activities.

The spatial proximity measure is used to compute a similarity matrix between all pairs of activities, which serves as input for the clustering, using the readily available implementation *hclust* in the statistical computing language *R*. We did not specify a maximum size or height of the clusters and determined the distance between two clusters by means of complete linkage. The details of the cluster assignments and merging steps are shown by the dendrogram in Fig. 2.

Next, we mine a reference component for each cluster by applying the trace-based RMM-2 approach. We apply the standard parametrization (noise level 0.2, minimum frequency 0.1) to ensure a sufficient frequency. Finally, diffusion and inflation rates are computed for each reference component. In Fig. 2, each cluster is annotated with its diffusion rate (top) and inflation rate (bottom). Ideally, both numbers should be close to 1. Lower diffusion and higher inflation rates indicate that merging two child components might not make sense. We see inflation rates smaller than 1, indicating that the merged component is smaller than the combined sizes of its children. This can be explained by control flow structures. For example, a self-loop with three nodes (one activity, two connectors) might be frequent in a larger log, but the connectors might not be contained in a merged component, e.g. a two-activity sequence.

The diffusion rates seem comparably low across all clusters. This can be attributed to the large log, where trace structures differ in frequency and variability. For example, the component where a loan is denied ("A_Denied, O_Refused")

Table 1. Identified Activities from the BPI 2017 log

A_Accepted	A_Submitted	O_Returned	W_Handle leads
A_Cancelled	A_Validating	O_Sent (mail and online)	W_Personal Loan collection
A_Complete	O_Accepted	O_Sent (online only)	W_Shortened completion
A_Concept	O_Cancelled	W_Assess potential fraud	W_Validate application
A_Denied	O_Create Offer	W_Call after offers	A_Create Application
A_Incomplete	O_Created	W_Call incomplete files	
A_Pending	O_Refused	W_Complete application	

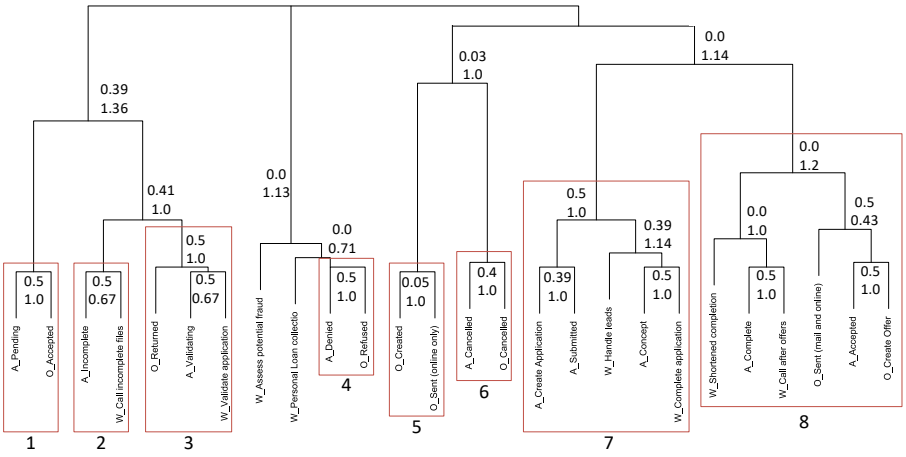

Fig. 2. Dendrogram for activity clustering

has low diffusion rates (rounded down to 0.0) from the lowest level on upwards, which suggests that it is not contained in a lot of traces. On the other hand, the component which describes the completion and submission of the application has fairly high diffusion rates. Based on the evaluation data, we selected eight reference components, marked by red borders in Fig. 2 and shown in Fig. 3.

Each selected reference component is a subprocess of the loan application process, with six small and two larger components. The latter are coherent subprocesses, namely the application creation (top) and application acceptance (bottom). The application creation appears straightforward. Offer creation and acceptance is kept separate, increasing its reusability, as it removes the strict association between applications and offers. The same applies to calling incomplete files; incomplete applications may be automatically denied. On the other hand, the subprocess is potentially incomplete and not directly usable. The second subprocess, application acceptance, is more specific to the individual process and therefore directly applicable, but less reusable. It describes the relation between offer and application, with the offer creation directly following the application acceptance. Counterintuitively, the offer creation is not part of this process, but this is not be supported by the clustering data.

Most small components describe the activities in a very specific situation, i.e. when an application is denied or canceled, if it is incomplete, or if the offer is accepted. These four components are applicable, but also fairly reusable due to their small size. They all associate offers with applications, which may impact their reusability. The remaining two components are the offer creation, which should be part of the application acceptance, and the application validation, which can be regarded as a subprocess in itself. The components' usability is impeded by back-loops, which often appear unnecessary (e.g. "O_Cancelled") and should be removed for a more generic component. Also, the lack of operators except XOR is apparent, as the components lack clear semantics.

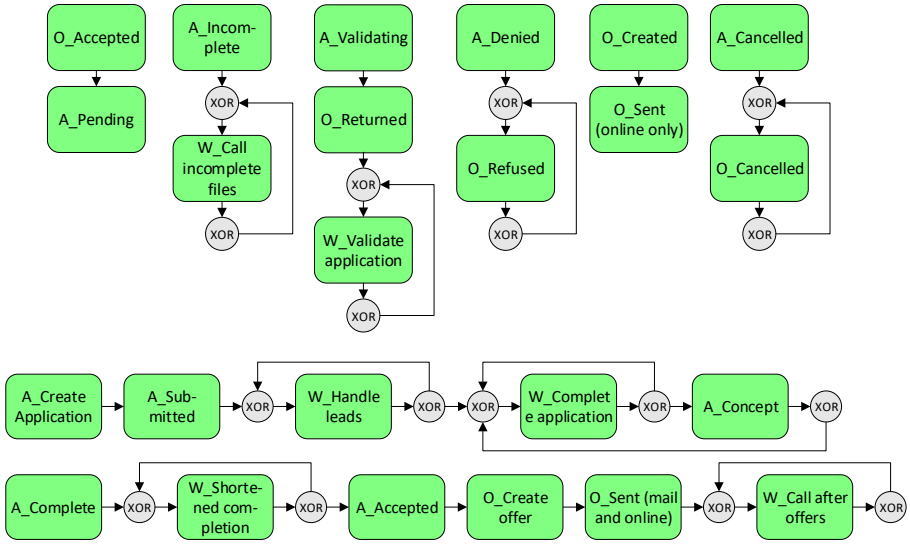

Fig. 3. Components mined for clusters 1-6 (top) and clusters 7 and 8 (bottom)

4 Related Work

In this contribution, we describe a technique to mine better reusable reference model components from instance-level event logs. Comparable approaches mine complete reference models, such as the approach by Gottschalk et al., which is explicitly set out to mine configurable reference models and according configurations from log files of well-running IT systems [14]. Instead of ensuring better reusability by mining smaller model components, the authors construct configurable reference models, which contain all potential process variants. This might work well, but a configurable model could become quite large, when covering rather diverse process behavior. Our own work on mining reference process models from large instance data works in a similar way, applying a clustering approach onto the traces of an execution log. The objective of this approach is to determine reference models depicting the whole process execution, but with differing degrees of generality or domain specificity [11], as opposed to reference components depicting process fragments. For this purpose, the event log is divided and clustered horizontally along the trace similarity instead of vertically according to activity proximity, as we do here.

All other automated approaches towards inductive reference modeling rely on type-level process models instead of instance-level events logs as input data. A contribution by Li et al. presents two approaches [15]. The first one uses a heuristic search algorithm and an approximation of the graph-edit distance for evolving an existing reference model such that it better fits a set of its derived variants. The second one uses an iterative clustering technique with a process-semantic proximity measure to construct a reference model from a predetermined

activity set. This is also closely related to our work here, however the goal is to combine all activities into a single reference model, solely based on their process-semantic order relations, while we are explicitly set out to mine smaller reference model components, based on an arbitrary proximity measure.

Throughout this article, we draw on the findings of several others areas of Business Process Management. For example, we determine the proximity between the activity pairs by means of a similarity measure. The more similar two activities are, the closer they should be associated within a cluster. Whereas we determine activity similarity based on spatial proximity, approaches to process model matching combine as many similarity measures as possible to determine the most appropriate correspondences between two process models [16]. These correspondences (called matches) are then used to determine the similarity between process models [17].

Clustering techniques are often used in BPM. One example is the identification of high-level activities from low-level event logs, which we apply as the first step in our approach. Günther et al. describe a technique that clusters events into recognizable patterns based on temporal and data-object-related proximity [7] as well as a new and improved technique, clustering event classes by means of trace segmentation [8]. This approach is very similar to ours in terms that it builds an event hierarchy solely based on a spatial proximity measure, but it operates on a much lower level of detail and is not directed towards the reusability of model components. Similarly, the POD-Discovery tool by Weber et al. also uses a hierarchical clustering approach to enable the application of process mining tools on low-level operational process logs [18].

5 Discussion and Conclusion

The contribution at hand has the objective to provide an automated and data-centered approach for supporting stakeholder in designing reference components. It is set out to address the dilemma of reusability, where the better a reference model applies to a situation, the fewer those situations are. We define reference components on spatial proximity, following the idea that semantically related activities are typically executed in close proximity to one another. This assumption can be questioned, as some empirical work suggests otherwise [19]. So, for more dependable results, other activity similarity measures, such as temporal proximity or label similarity could also be taken into account.

Generally, the reference component's domain and characterization are determined by the input data, which, in turn, depends on the intended usage of the reference components. If considering data from one company and one process only, the components will be process-specific, i.e. focus on patterns within this process. Considering the same process across multiple companies discovers cross-organizational similarities or patterns, i.e. industry-specific components. Organization-specific components describe similarities across multiple processes, i.e. common process patterns in one company. Finally, when analyzing data from multiple processes and multiple companies, we obtain generic, domain-

independent components. All of these scenarios have meaningful applications, but this choice should be considered prior to determining the components, as it will influence the appropriate values for diffusion and inflation rate.

One could also argue that the input data needs to be carefully chosen with regard to the represented process. The main motivation of this article is the dilemma of reusability, i.e. the difficulties that appear when reusing models or model parts in the design of a new conceptual model. There is no point in designing reference components for a domain in which no subprocess is sufficiently frequent or relevant such that there is a general interest in reusing it. This means, that repetitive processes, like the loan application process in our case study, are particularly suitable for such an analysis. This is also evident, as those processes are most likely supported by information systems, such that process logs exist.

While it is also possible to mine reference components from type-level model data, we have decided to base our approach on instance-level execution data, which offers a more realistic perspective on the process than type-level data. By computing the activity proximity based on factually executed process traces associated with timestamps, resources, or related data objects, we are able to provide a realistic view on the process that is actually executed, increasing the probability for component reuse. However, this means that our result is a set of *descriptive* reference components instead of *normative* ones. They depict as-is process behavior instead of to-be recommendations. Both have realistic application scenarios. Companies typically strive to standardize support processes, such that more resources are available to focus on their core processes as a competitive advantage. Hence, the former, which are often automated and IT-supported, can be designed by reusing a descriptive reference model.

As reference model design always depends on the intended (re-)use and purpose of the models in the given domain, our objective is not to design a completely automated approach, but rather a helpful tool for decision-making support. Full automation is mainly difficult, because constructing completely generic components requires some abstraction of either activities or the entire process model (e.g. by changing the label to describe a more generic task like “check document” instead of “check invoice”). Those abstractions are context-dependent and thus comparably hard to find by algorithmic means, but fairly easy for a human process designer to make.

References

1. Becker, J., Meise, V.: Strategy and organizational frame. In Becker, J., Kugeler, M., Rosemann, M., eds.: Process Management. A Guide for the Design of Business Processes. Springer (2011) 91–132
2. Fettke, P., Loos, P.: Perspectives on reference modeling. In Fettke, P., Loos, P., eds.: Reference Modeling for Business Systems Analysis. Idea Group Publishing (2007) 1–20
3. Rupprecht, C., Fünffinger, M., Knublauch, H., Rose, T.: Capture and dissemination of experience about the construction of engineering processes. In Wangler,

B., Bergman, L., eds.: 12th International Conference on Advanced Information Systems Engineering, June 5–9, Stockholm, Sweden, Springer (2000) 294–308
4. Becker, J., Pfeiffer, D., Räckers, M.: Domain specific process modelling in public administrations – the picture-approach. In Wimmer, M.A., Scholl, J., Grönlund, Å., eds.: 6th International Conference on Electronic Government, September 3-7, Regensburg, Germany, Springer (2007) 68–79
5. Dadashnia, S., Fettke, P., Hake, P., Klein, S., et al.: Exploring the potentials of artificial intelligence techniques for business process analysis (2017) http://www.win.tue.nl/bpi/lib/exe/fetch.php?media=2017:bpi2017_paper_41.pdf.
6. van der Aalst, W.: Process Mining: Data Science in Action. 2nd edn. Springer (2016)
7. Günther, C., van der Aalst, W.: Mining activity clusters from low-level event logs. BETA Working Paper Series, WP 165, Eindhoven University of Technology, Eindhoven (2006)
8. Günther, C.W., Rozinat, A., van der Aalst, W.M.P.: Activity mining by global trace segmentation. In Rinderle-Ma, S., Sadiq, S., Leymann, F., eds.: Business Process Management Workshops, Ulm, Germany, September 7, 2009. Revised Papers, Springer (2010) 128–139
9. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P., Toussaint, P.J.: From low-level events to activities - a pattern-based approach. In La Rosa, M., Loos, P., Pastor, O., eds.: 14th International Conference on Business Process Management, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings, Springer (2016) 125–141
10. Everitt, B.S., Landau, S., Leese, M., Stahl, D.: Hierarchical Clustering. In: Cluster Analysis. John Wiley & Sons, Ltd (2011) 71–110
11. Rehse, J.R., Fettke, P.: Mining reference process models from large instance data. In Dumas, M., Fantinato, M., eds.: Business Process Management Workshops, Rio de Janeiro, Brazil, September 19, 2016, Revised Papers, Springer (2017) 11–22
12. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer (2007)
13. Van Dongen, B.: BPI challenge 2017 https://data.4tu.nl/repository/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b.
14. Gottschalk, F., van der Aalst, W., Jansen-Vullers, M.: Mining reference process models and their configurations. In: On the Move to Meaningful Internet Systems: OTM 2008 Workshops, Springer (2008) 263–272
15. Li, C., Reichert, M., Wombacher, A.: Mining business process variants: Challenges, scenarios, algorithms. Data & Knowledge Engineering **70**(5) (2011) 409–434
16. Antunes, G., Bakhshandelh, M., Borbinha, J., Cardoso, J., Dadashnia, S., et al.: The process matching contest 2015. In Kolb, J., Leopold, H., Mendling, J., eds.: Proceedings of the 6th International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA-15), September 3-4, Innsbruck, Austria, Köllen Druck+Verlag GmbH, Bonn (9 2015)
17. Becker, M., Laue, R.: A comparative survey of business process similarity measures. Computers in Industry **63**(2) (2012) 148–167
18. Weber, I., Li, C., Bass, L., Xu, X., Zhu, L.: Discovering and visualizing operations processes with POD-Discovery and POD-Viz. In: 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE (2015) 537–544
19. Klinkmüller, C., Weber, I.: Analyzing control flow information to improve the effectiveness of process model matching techniques. Decision Support Systems **100** (2017) 6–14