Information Systems 84 (2019) 63–87

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

Semi-automatic inductive construction of reference process models that represent best practices in public administrations: A method

Hendrik Scholta [a,*], Marco Niemann [a], Patrick Delfmann [b], Michael Räckers [a], Jörg Becker [a]

[a] University of Muenster - ERCIS, Leonardo-Campus 3, 48149 Muenster, Germany
[b] University of Koblenz–Landau, Universitätsstraße 1, 56070 Koblenz, Germany

HIGHLIGHTS

- Inductive reference modeling (IRM) is regarded supportive for process management.
- Current IRM methods consider common practices in reference models only.
- We build the first IRM method that also allows to detect and include best practices.
- We conceptualize the method and demonstrate its functionality using an example.
- We implement the method and provide a workshop-based evaluation.

ARTICLE INFO

Article history:
Received 23 February 2018
Received in revised form 23 February 2019
Accepted 4 March 2019
Available online 20 March 2019
Recommended by Matthias Weidlich

Keywords:
Process management
Process modeling
Reference modeling
Process model merge
E-government
Public administration
Benchmarking
Model querying

ABSTRACT

Business process management often uses reference models to improve processes or as starting point when creating individual process models. The current academic literature offers primarily deductive methods with which to develop these reference models, although some methods develop reference models inductively from a set of individual process models, focusing on deriving and representing common practices. However, there is no inductive method with which to detect best practices and represent them in a reference model. This paper addresses this research gap by proposing a method by which to develop reference process models that represent best practices in public administrations semi-automatically and inductively. The method uses a merged model that retains the structure of the source models while detecting their common parts. It identifies best practices using query constructs and ranking criteria to group the source models' elements and to evaluate these groups. We provide a conceptualization of the method and demonstrate its functionality using an artificial example. We describe our implementation of the method in a software prototype and report on its evaluation in a workshop with domain and method experts who applied the method to real-world process models.

1. Introduction

Public administrations have increasingly recognized the need to improve their internal processes [1,2]. One way to improve processes is to align existing processes with best practices, represented by reference process models [3], which are "generic conceptual models that formalise recommended practices for a certain domain" [3, p. 595]. Reference models may also be used as a basis in a modeling process and adapted to a specific organization afterward instead of creating a process model from scratch, decreasing modeling time and cost [3,4].

A considerable amount of research has focused on reference process modeling in general [5–10] and in the domain of public administrations in particular [11–13]. Both deductive and inductive strategies are used in designing reference models [14], where deductive methods create reference models by specializing general theories and concepts, and inductive methods generalize individual models by abstracting from unnecessary details. The literature has offered a few inductive methods, but most of the strategies for reference model construction are deductive in nature [15,16].

Despite their rare use, inductive methods are relevant to business process management since individual models are operationalizations of abstract guidelines like laws or business rules. For instance, laws specify the requirements an applicant must meet to receive a service, and public administrations check these requirements in the course of service delivery. Process models concretize such laws by specifying how to check the requirements

* Corresponding author.
E-mail addresses: hendrik.scholta@ercis.uni-muenster.de (H. Scholta), marco.niemann@ercis.uni-muenster.de (M. Niemann), delfmann@uni-koblenz.de (P. Delfmann), michael.raeckers@ercis.uni-muenster.de (M. Räckers), joerg.becker@ercis.uni-muenster.de (J. Becker).

https://doi.org/10.1016/j.is.2019.03.001
0306-4379/

and in what order. Inductive methods provide empirical evidence and are realistic solutions to organizational problems, as has been shown in practice. In contrast, there is no guarantee of the practical feasibility of deductively developed reference models. While many reference models have been developed at least partly inductively [16], and several automatic or semi-automatic inductive methods have recently emerged that focus on deriving and representing common practices using reference models (e.g., [17–19]), the use of induction to develop reference process models that represent *best practices* has not yet been investigated. In particular, no methods have been developed that consider the definition of "best" depending on the application context. Therefore, focusing on the specific domain of public administrations, this paper's research goal is to *design a method that semi-automatically constructs from a set of individual process models reference process models that represent best practices in public administrations*.

The method designed here, called *RefPA* (a combination of "reference model" and "public administration"), returns a reference model from individual models that various institutions have created using the same modeling language and that deal with the same process. The paper refers to the individual models that serve as a basis for the reference model's construction as *source models*.

The RefPA method uses three main steps to detect and represent best practices in process models. The *preprocessing step* merges all source models into a single model that defines the reference model's structural foundation. We recommend using one of the many methods that have been defined for merging process models, rather than defining yet another model-merge method for this step. Then, in the *first step*, the elements that are common to all source models are considered best practices and retained in the merged model, as all models use them to provide an equal solution. In the *second step*, we identify best practices for tasks that have differing solutions in the source models by formulating queries for the source models and then comparing their model segments according to *best practice criteria* that we identified by means of an extensive literature study. The segments that perform best remain in the merged model. Thus, this paper's main contribution is a specialized *query language to detect best practices* in process models that should be combined in a reference model. This query language differs from existing model-query languages, as it allows model elements to be grouped to form model segments and such groups to be evaluated to detect best practices.

However, the notion of *best practices* in this paper is limited to the set of available source models. As RefPA operates only on the source models, it detects only the best practices that are in the source models, not best practices that are theoretically possible and theoretically superior but are not in the source models. Therefore, we cannot assume that no solution is possible that is better than the solutions that are available in the source models. In addition, the definition of best practices always depends on the user's perception. For example, some people seek a low-cost solution regardless of quality, while others seek a high-quality solution, regardless of cost. Hence, RefPA does not provide objective best practices but makes suggestions for subjective best practices that the user evaluates based on his or her expertise and goal. Consequently, users can interpret RefPA's results as suggestions for *relative and perceived best practices*.

RefPA can be used to develop a reference model that is dedicated to public administrations in *general*. The context of public administrations is especially suitable for creating inductive reference models as, in contrast to private organizations, public administrations tend toward collaboration instead of competition [20]. The tendency to collaborate facilitates the sharing of process models for the same tasks and services, which is necessary in consolidating several process models from multiple organizations into reference models. Process model libraries that share process knowledge across public administrations can help reference modelers to implement the RefPA method [21,22] by acquiring as many source models as possible to increase the probability that the set of source models will include not only relative but also absolute best practices. Excess source models that do not contain best practices do not negatively influence the resulting reference model, as their practices are not incorporated into the reference model.

RefPA can also be applied to the development of a reference model that is relevant to a *specific* context. Since the same legal regulations define the services and processes of numerous public administrations and limit their flexibility [20], process managers can use RefPA in process-harmonization scenarios to increase efficiency. Public administrations that introduce a commonly used software system can implement a reference process model that uses a set of source models that the public administrations that participate in the project define.

This paper's research methodology is based on the design science research paradigm [23] and the design science research methodology (DSRM) presented by Peffers et al. [24]. Since we seek to design a practical IT artifact, we define the research gap (step 1, as above), derive the solution's requirements and objective and discuss related work (step 2, in Section 2), develop the artifact – the RefPA method – (step 3, in Section 3), demonstrate the method's functionality by applying it to an artificial example (step 4, in Section 4), and present the results of our evaluation (step 5, in Section 5). Finally, we provide a conclusion and suggestions for future work in Section 6. We also conducted classic software engineering to implement our IT artifact and evaluated the IT artifact in a workshop in which the prototype was used and tested with domain and method experts. Data for the evaluation was collected based on responses from workshop participants to a questionnaire that assessed our IT artifact in the spirit of the DSRM.

2. Foundations and research gap

2.1. Requirements

This section presents the requirements for a method that semi-automatically and inductively constructs reference process models that represent best practices. We derived the requirements based on the literature and our expertise and experience in research and practice but do not claim that the list of requirements is complete. Our derivation of requirements revealed that, to meet the goal of *detecting* and *representing* best practices, RefPA must meet four functional and one non-functional requirements.

(a) Functional Requirements (FRQs)

FRQ1: Segmentation. To *detect* best practices, an inductive reference modeling method must divide the source models into segments, as not all best practices are contained in a single source model. Therefore, segments of the source models, not entire models, constitute best practices. Model segments are subgraphs with nodes and edges, so the solution must form groups of nodes and edges in each source model. As model segments are solutions for certain tasks, and the tasks that are relevant to a reference model depend on the purpose of the reference model and, thus, vary from case to case [25,26], the user must define what constitutes the groups. Consequently, the solution must offer a mechanism that allows the user to segment the source models into groups.

FRQ2: Evaluation. Since the goal is to *detect* best practices and detecting the "best" implies a ranking [27], the groups mentioned

in *FRQ1* must be compared, evaluated, and ranked. As the definition of "best" (e.g., fastest, cheapest) depends on the reference model's goal, the user must specify the parameters for the evaluation. Therefore, an inductive reference modeling method must have a repository that contains various interpretations of "best" with which to evaluate groups. Since public administrations and private companies differ [28], the solution must be tailored to the appropriate domain. To be precise, the solution should also use the semantic information of process model activities, such as roles and software systems. Consequently, the solution must provide a mechanism that allows the user to evaluate and rank groups.

FRQ3: Integration. To *represent* best practices, the solution integrates groups that depict best practices into a single reference model. Because of its inductive nature, the method produces a reference model by integrating the source models, which is the constituting characteristic of all inductive methods [14]. Thus, the source models predefine the structure of the resulting reference model, which contains activities and control-flow relationships that are present in at least one source model. However, since not all of the source models' elements constitute best practices, the solution incorporates only those activities and control-flow relationships that constitute best practices.

FRQ4: Modeling Language. To support a wide range of scenarios and achieve a high degree of generalizability, a solution cannot be limited to a certain process modeling language but must be able to operate on process models regardless of their modeling language. However, all of the source models must share the same modeling language. Since public administrations use multiple modeling languages, such generality is necessary if the reference model is to be adopted regularly in practice [29]. Consequently, a solution must rely on a formalization that fits the characteristics of process modeling languages in general.

(b) Non-Functional Requirements (NFRQs)

NFRQ1: Applicability. Since a reference model has a wide range of applications, a large variety of potential users, including IT experts but also process managers in public administrations who may not have comprehensive IT knowledge, should be able to apply the solution. Consequently, a solution must have a high perceived ease of use, that is, a high "degree to which a person believes that using a particular system would be free of effort" [30, p. 320]. A system or method has a high perceived ease of use if it is easy to learn, controllable, clear and understandable, flexible, easy to master, and easy to use [30].

2.2. Related work

Before outlining our RefPA method in Section 3, we take a step back and analyze prior research to describe the state of the art and the existing research gaps with regard to the five requirements. We sought research on methods for creating reference process models inductively and proposed languages for querying process models. The assessment distinguished three degrees of fulfillment, +, o, and -, per requirement, as shown in Table 1. Only the FRQs are addressed in Table 1, as NFRQs would require a different mode of analysis because of their subjectivity and relativity [31].

2.2.1. Methods for the inductive creation of reference process models

While this paper develops a novel inductive method with which to create reference process models, many contributions have been made to inductive reference process modeling in the last decade, which range from largely generic procedure models that focus, for instance, on selecting appropriate modeling techniques based on situational contexts [32] to fully algorithmic methods that, for instance, use genetic algorithms [33] or factor analysis [17]. Our non-exhaustive sample even includes one publication [34] that was originally set in the process-model-merging domain and, through the application of thresholds, represents a form of inductive reference process modeling.

The results of our assessments are summarized in Table 2. We selected a first set of inductive methods based on our knowledge as experts in the field. We complemented this initial set of methods using the results of a keyword-based search with keywords like "Inductive", "Reference Model", "Development", and "Construction", and a subsequent backward and forward search of the resulting articles. As Table 2 shows, none of the existing methods covers all of the requirements for the detection and representation of best practices. Especially in terms of segmenting processes and evaluating these segments, which are necessary to detect "best" process parts, many of the extant work has fallen short, as they typically provide reference models that represent "common" practices. The subsequent paragraphs outline our reasoning behind the assessments presented in Table 2.

Algorithmic Methods

The first subgroup of related methods we analyzed are algorithmic methods. Similar to the proposed RefPA method, they all provide a full algorithmic description of how to elicit a reference model inductively from a set of input process models.

Almost all members of this class of methods fulfill the integration requirement, *FRQ3*. Considering that all methods seek to derive a reference model based on a set of source models, the high degree of fulfillment is not surprising. However, the methods contain significant differences in how *Integration* is handled. One set of algorithms approaches the task of assembling the reference model by iteratively including activities and flow relationships, omitting any form of upstream segmentation. A typical example is Yahya et al. [40], who base their method on the so-called activity proximity score (APS), a measure that indicates the average proximity of two activities across a set of processes, and build their reference model by determining a start activity and then adding the activities with the highest APS. Other algorithms, such as the one based on genetic algorithms (GAs) Yahya et al. [41] propose, encode activities and flow relationships in a numeric "genome", apply the GAs and then set up the reference model by decoding the genome into a graph structure again. Other authors, such as Rehse et al. [16], take an *Integration* approach similar to the RefPA approach described in Section 3 (especially Sections 3.3 and 3.6) by segmenting the source models and recombining the reference model from a selected set of segments. However, one algorithmic method, that of Martens et al. [17], does not address *FRQ3*, as instead of reassembling common – or best – parts from the source models, they perform a factor and cluster analysis based on the adjacency matrix to identify the source model that is closest to the intended reference process model and then fine-tune it by removing the edges and nodes that are not sufficiently common.

Another requirement that stands out is *FRQ2*, which deals with the *Evaluation* of activities and groups. While none of the methods in the extant literature totally fulfills the requirement, all methods do so at least partially. Considering that a reference model, in contrast to, for example, a merged model, typically has at least some form of filtering to detect the commonalities or best practices, the degree of fulfillment observed is not surprising. The simplest form of evaluation used in the single methods is frequency-based thresholding. For example, La Rosa et al. [34] use the frequency of flow relationships across the source models to remove those that are not sufficiently common. Similarly, Rehse et al.'s [16] method uses frequency to identify subgraphs that should be integrated into the reference model, while Li et al. [36]

Table 1
Evaluation criteria for the assessment of existing methods.

	−	O	+
FRQ1:	No segmentation	Segmentation without user input	User-parameterized segmentation
FRQ2:	No evaluation of groups/single activities	Some form of evaluation of groups/single activities	User-parameterized evaluation of groups/single activities, including ranking
FRQ3:	No assembly of a reference model		Assembly of some form of reference model
FRQ4:	≤1 language	≥ 1 language (no clear statement regarding whether more than one language is possible)	>1 language

Table 2
Comparison of RefPA to related methods.

	Method	FRQ1 (Segmentation)	FRQ2 (Evaluation)	FRQ3 (Integration)	FRQ4 (Modeling Language)
Algorithmic Methods	[18]	o	o	+	−
	[34][a]	o	o	+	−
	[35]	o	o	+	−
	[36][b]	o	o	+	o
	[33]	o	o	+	o
	[17]	−	o	−	+
	[37]	−	o	+	+
	[38]	−	o	+	−
	[16]	o	o	+	+
	[19]	−	o	+	−
	[39]	−	o	+	o
	[40]	−	o	+	o
	[41]	−	o	+	o
Generic Procedure Models	[42]	o	o	+	+
	[32]	−[c]	−	−	+
	[43]	−	−	−	+
	[44]	−	−	+	−
	RefPA	+	+	+	+

[a]Two methods are presented; the first creates a merged model and the second derives a reference model. Since they build on each other, they are considered one method.
[b]Of the two methods proposed (configuring an existing reference model; discovering a reference model), only the discovery method is relevant to our context.
[c]The paper does not discuss *FRQ*1, *FRQ*2, or *FRQ*3 since it does not propose a novel method but gives advice on how to choose a valid technique.

again use activity frequencies to evaluate which activities to transfer into a reference model. The remaining methods use other and often more sophisticated evaluation strategies, especially the methods that use GAs and evolutionary strategies [19,33,39,41]. For example, Sonntag et al. [19] generate their reference models not based on the source models' graph structures but on so-called performer networks (PNs) (the social networks of the agents who perform the business processes). Hence, they use an evolutionary strategy to create network solutions and evaluate them with a fitness function that determines the networks' efficiency toward each input model to select the best PN and derive the final reference model. Yahya et al. [39] go farther in evaluating their GAs' genomes based on multiple objectives (maximum proximity score, as defined in [45], cost, and duration) that are restricted by an even larger set of constraints (e.g., start nodes that have no predecessors). As a result of using multi-objective optimization, each evaluation returns multiple "best" solutions, although without giving the user the chance to parameterize the evaluation further.

While *FRQ2* and *FRQ3* are at least partially fulfilled by almost all of the related methods analyzed here, *FRQ1* sorts out half of the methods since they support neither user-parameterized segmentation nor segmentation in general. However, even among the methods that use segmentation, the exact type of segmentation differs. Some authors, such as Leng and Jiang [35], determine the longest common subsequences within the source models and used them to assemble the final reference model. Others (e.g., Ardalani et al. [18] and Rehse et al. [16]) are even more restrictive, limiting their segments to two and one consecutive elements, respectively, to which they refer as a "singleton subgraph" [16]. Hence, among the few methods that at least segment the source models into common – not best – subsequences, some severely restrict the potential segments based on computational complexity [16]. Other methods (e.g., [41]) refrain completely from using segments and consider only single activities and links in the process-representing genomes from which their final reference models are derived item by item. Martens et al. [17] do not even subdivide their source models into activities and flow relationships but use factor and cluster analysis to identify the most suitable reference model and refine it.

Compared to the other three requirements, *FRQ4* is the most diverse in terms of degree of fulfillment. A group of researchers around Fettke and Loos proposes several methods that are explicitly tailored to work with event-driven process chains (EPC) [18,19,38], while authors like Leng and Jiang [35] focus on a specific representation of service processes. Other studies, such as [33] and [40], are less restrictive but also more ambiguous regarding the modeling language used, as they provide formal or informal specifications of how business process models should look and do not explicitly map these specifications to a specific notation. However, both papers use examples and hints ("exist, e.g. in EPCs" [33, p. 4]), which makes it difficult to determine their methods' generalizability. On the other hand, some authors are almost generic regarding their supported modeling languages. For example, Rehse et al. [16] formally define processes in a way that supports several notations, including EPC and BPMN, but point out that their method does not support gateway-less modeling languages. Others, such as Martens et al. [17, p. 442], go even

farther in fulfilling *FRQ4* by explicitly providing a method that is "independent from a concrete modeling language".

Generic Procedure Models

Several more generic publications with regard to the creation of reference process models range from procedure models on how to create reference models – yet with no specific algorithmic or implementation details – [42] to even more generic methods that propose and use a reference process-mining technique that is based only on a given context [32].

Table 2 shows several differences in these methods in comparison to the algorithmic methods. For example, almost all of the algorithmic methods fulfill *FRQ3*, only half of the generic methods achieve that degree of fulfillment. The two methods in [42] and [44] outline a comparatively specific procedure model on how to establish a reference model to fulfill *FRQ3* since both point out the necessity to assemble a reference model from elements of the source models. The two methods that do not fulfill the requirement are those that can be configured with specific mining techniques, which then determine how the reference model is assembled [32,43].

The generic methods' difference from the algorithmic methods is even more striking in *FRQ2*, as only one of the more generic methods even partially fulfills the requirement. Even though [42] does not provide a specific algorithm, it points out that suitable elements for a reference process model must be chosen on the basis of sufficient commonality (frequency of occurrence in source models). Other methods, such as [32] and [43], at best mention that an evaluation may be required or possible but leave the final decision and specification to the reference process-mining technique chosen.

Similarly, [42] is the only method that explicitly mentions the need for segmentation of subgraphs that are sufficiently common among the source models. The other methods either refrain from mentioning *Segmentation* at all [44] or leave the "if" and "how" of *FRQ1* to a method that is chosen as part of the proposed method [32,43].

FRQ4 is the one requirement for which the generic methods outperform the algorithmic methods. Without giving exact algorithmic descriptions, all methods except one that focuses on Workflow Nets [44], are sufficiently generic to be implemented in a wide variety of process modeling languages. The examples and evaluations of [32,42,43] are based primarily on EPC process models, given that the authors are part of the group of researchers around Fettke and Loos, but they do not restrict themselves to that modeling language.

This analysis makes apparent that none of the techniques proposed to create reference models inductively can be used to establish best practice models based on requirements *FRQ1*, *FRQ2*, *FRQ3*, and *FRQ4*. While many of the methods we evaluated, as expected, fulfill *Integration*, most struggle with either *Segmentation* or *Evaluation* in general or at least when it comes to the parameterizable *Segmentation* and *Evaluation* to identify what is "best" for a given context and modeling goal.

2.2.2. Query languages for process models

One of the weak spots we identified is the weak level of fulfillment of *FRQ1* and *FRQ2*, so before proposing an entirely novel method with which to create best practice reference models inductively, we looked at the domain of process model query languages to find candidates for eliciting process segments that could be extended to become methods for creating reference process models. While many of the commonly stated definitions implicitly or explicitly refer to the purpose of query languages in the discovery and retrieval of process models that satisfy a user-defined (e.g., textual or graph-structural queries as one form of input) set of requirements [46–48], others suggest their applicability to more analytical purposes, such as compliance, checking for patterns of weakness [49], and several more [50].

The domain of query languages for process models and process model repositories is largely mature, so we worked through several recently published, major studies on process model query languages [46,47,49–51] but did not find a suitable method for reuse in the creation of reference process models. As expected, none of the languages fulfills *FRQ3*. However, since the purpose of model-querying languages is to elicit information from existing models, not to compose novel models, this criterion can be neglected for the most part. While the ability to extract groups of process model elements is one of the more common functionalities of most query languages, such studies as those of Momotko and Subieta [52] and Di Francescomarino and Tonella [53] show that this ability is not guaranteed. However, a large number of query languages are still viable in our context since they provide textual [49] or visual [54,55] constructs to find and extract process model segments from a larger process model, fulfilling *FRQ1*. Another hurdle many of the query languages fail to clear is *FRQ4*, that is, the ability to support multiple *Modeling Languages*. Here again, many of the proposed languages use components that are specific to one modeling language (e.g., BPMN [52,55] or BPEL [56]), but even those few languages that fulfill *FRQ1* and *FRQ4* fail to fulfill *FRQ2*. While, GMQL [49] allows the attributes of single activities to be queried, it does not provide functionality needed to evaluate selected subgraphs based on a set of user-defined parameters or even to rank them according to some default criteria.

Therefore, while many process model query languages are promising alternatives with regard to *FRQ1*, no candidate can serve as the basis for the creation of reference process models. Combining these insights with those from the prior subsection, we conclude that, while several methods create reference process models inductively, no method has been proposed to create *best practice* reference process models, as specified by *FRQ1*, *FRQ2*, *FRQ3*, and *FRQ4*.

The following section introduces the novel RefPA method, which fulfills all four requirements and supports process managers in establishing best practice reference process models.

3. The method

3.1. Overview

RefPA's steps are visualized in Fig. 1. Steps with a user icon at the top-right corner require user interventions, whereas steps without a user icon are executed automatically. Fig. 1 inputs three source models and their merged model to RefPA (step 0). The merged model contains all of the source models' nodes and edges and serves as basis for the reference model. Since the reference model emerges from the merged model, RefPA uses the merged model to fulfill *FRQ3*, and since the reference model evolves from the merged model by selecting and marking suitable elements, the merged model determines the positions of the elements in the reference model and the reference model keeps the structure of the source models.

To identity best practices, that is, the elements that need to be marked to be transferred to the reference model, RefPA first detects common parts of the source models and then processes non-common parts. Best practices are the elements all source models agree are the only solution to a certain task in the process. RefPA discovers segments that all source models have in common (*FRQ1*) and marks them for the reference model (*FRQ3*). Further evaluation of these segments is not necessary (*FRQ2*) since there are no alternative solutions available in the solution space.

RefPA uses the merged model and its elements' references to their source elements to identify automatically the elements

that all of the source models use (step 1). If an element of the merged model is common to all source models, then it has a corresponding element in each source model, that is, the number of elements it refers to is equal to the number of source models. In Fig. 1, the number of referred elements is exemplarily annotated next to each node. Since node A refers to three source elements and there are three source models, A is a common element. On the other hand, C has two source elements and is not a common element (2 < 3). To incorporate all common process segments into the reference model, common elements are automatically marked and then transferred to the reference model in step 4. In Fig. 1, marked nodes and edges are highlighted by bold text and lines.

RefPA must also detect the best solutions to tasks when the source models' solutions differ. To process these non-common parts, the user can define queries that are automatically executed in the source models to group model elements (*FRQ1*) and evaluate these groups (*FRQ2*) in steps 2 and 3. Such queries detect best practices since they evaluate the source models' various solutions to tasks based on best-practice criteria from the literature. The grouping identifies "practices" and the evaluation identifies "best practices". For example, the construct `GROUP BY` can be used to specify how the elements are grouped, and `ORDER BY` can be applied to specify how the groups are evaluated. The query constructs build on the query language SQL to ensure a high level of applicability (*NFRQ1*). The result of each query is an overview of the ranking of the groups. The groups ranked at the top (Groups II.1 and III.2 in Fig. 1) represent best practices, so the user can mark these groups to be transferred to the reference model.

Finally, in step 4, the reference model is assembled (*FRQ3*) by automatically removing nodes and edges from the merged model that were not marked when the common parts were processed in step 1 or when the non-common parts were processed in steps 2 and 3 (e.g., node C in Fig. 1). The user must ensure that the reference model is connected and syntactically correct by, for example, removing gateways if they have only one ingoing and one outgoing edge. The user can rely on functionalities that have been implemented in existing process modeling tools to check the syntactic correctness of the resulting reference model.

The following subsections formalize and explain the steps in detail. A list of symbols is provided in Appendix A.

3.2. Step 0: Source models

RefPA receives several process models as input. Based on [57], and to meet *FRQ4*, we define a process model m_i generally as tuple $m_i = (A_i, B_i, F_i, P_i, V_i, props_i)$ with $1 \leq i \leq |S|$ and

- A_i: A finite, non-empty set of activities
- B_i: A finite set of gateways
- $N_i = A_i \cup B_i$: The set of nodes
- $F_i \subseteq N_i \times N_i$: The set of flow relationships (i.e., edges)
- P_i: A finite, non-empty set of properties
- V_i: A finite, non-empty set of all occurring property values
- $props_i: A_i \times P_i \rightarrow V_i$ is a mapping that assigns a property value to a pair of one activity and one property.

$S = \{m_i\}$ denotes the set of source models. The property values in V_i are sets used to, for instance, assign more than one data object to an activity. Therefore, V_i is a set of sets.

Typical properties that are commonly used in process models (e.g., [58–60]) include:

- Caption
- Data Object
- IT System
- Role
- External Stakeholder
- Legal Regulation
- Cost
- Time
- Probability

We use these properties to illustrate how RefPA works, although this list is not exhaustive. One potential extension of the list could be a differentiation between input data objects and output data objects, where RefPA could refer to the data object and the ingoing or outgoing edge as one property. In general, all properties of the activities provided by the process modeling language used can be referred to in RefPA queries. When applying RefPA, the user decides which of the properties provided by the respective modeling language are to be accessed in the queries.

In the subsequent sections, we transform a tuple into a set. For this purpose, we define the function $Set\left(\left(x_j\right)\right) = \{x_j | x_j \in \left(x_j\right)\}$.

3.3. Step 0: Merged model; step 1: Detection of common parts

In addition to the source models, a merged model that consolidates the source models and is created manually or using one of the existing algorithms is input to RefPA: $m_m = (A_m, B_m, F_m, P_m, V_m, props_m)$. Existing merge algorithms rely on mappings of the source models' elements. In RefPA's subsequent steps, we use such mappings that consist of the functions *s_n* and *s_e*:

- The function $s\_n: N_m \rightarrow N_1 \times N_2 \times \cdots \times N_{|S|}$ maps each node of N_m to its counterparts in the source models. For each $n_m \in N_m$ there is at least one mapped counterpart in at least one source model and at most one counterpart in each source model. If there is no counterpart in one of the source models, then the corresponding part of the tuple $s\_n(n_m)$ is empty. $s\_n\left(n_{mj}\right) = \left(n_{1j}, n_{2j}, \ldots, n_{|S|j}\right), |Set(s\_n\left(n_m\right))| \in \{1, \ldots, |S|\}, j \in \{1, \ldots, |N_m|\}$. In addition, for each source node per source model, there is exactly one counterpart in the merged model, that is, $\forall k \in \{1, \ldots, |S|\}, l \in \{1, \ldots, |N_k|\}$ $\exists! n_{mj} \in N_m : n_{kl} \in Set(s\_n(n_{mj}))$.
- Similarly, the function $s\_e: F_m \rightarrow F_1 \times F_2 \times \cdots \times F_{|S|}$ refers to the sources of a merged model's edge. For each $f_m \in F_m$ there is at least one mapped counterpart in at least one source model and at most one counterpart in each source model. If there is no counterpart in one of the source models, then the corresponding part of the tuple $s\_e(f_m)$ is empty. $s\_e\left(f_{mj}\right) = \left(f_{1j}, f_{2j}, \ldots, f_{|S|j}\right), |Set(s\_e\left(f_m\right))| \in \{1, \ldots, |S|\}, j \in \{1, \ldots, |N_m|\}$. Moreover, for each source edge per source model, there is exactly one merged edge, that is, $\forall k \in \{1, \ldots, |S|\}, l \in \{1, \ldots, |F_k|\}$ $\exists! f_{mj} \in F_m : f_{kl} \in Set(s\_e(f_{mj}))$.

Since the reference model emerges from the merged model, we define a function $m\_n: N_m \rightarrow \{0, 1\}$ that indicates whether a merged node has been marked to be transferred to the reference model ($m\_n(n_m) = 1$) or not ($m\_n(n_m) = 0$). Similarly, we define $m\_e: F_m \rightarrow \{0, 1\}$.

There can be at most one node per source model that refers to a merged node; that is, a source node can be mapped to at most only one other node of each source model. We restrict the method to 1:1 mappings since otherwise merged nodes could be marked even if the user selects only a small fraction of the merged node in steps 2 and 3. For example, if a merged activity refers to twenty activities in a first model and one activity in a second model, the entire activity would have to be marked if the user selects a group with only one of the activities of the first model, even though only one-twentieth of its complexity affected the group.

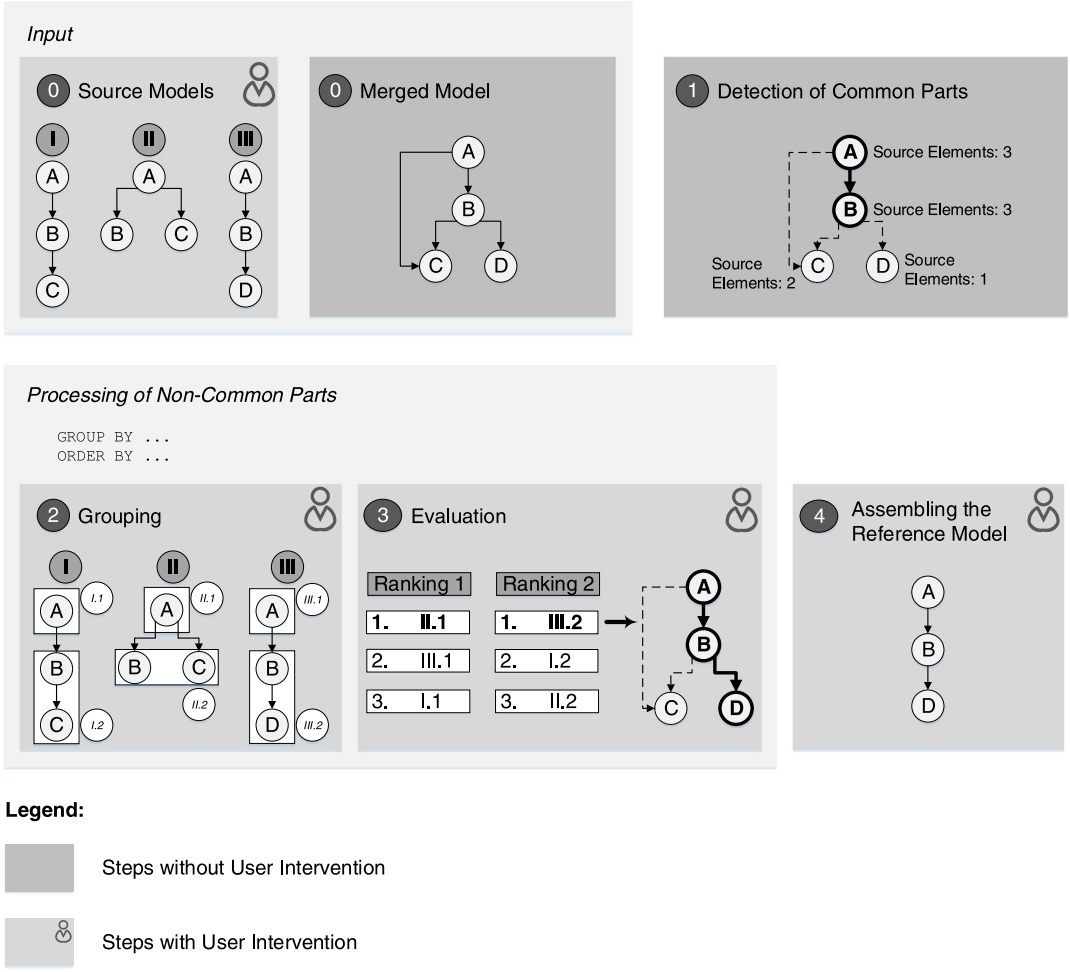

Fig. 1. Steps of the proposed method.

All nodes $n_m \in N_m$ and edges $f_m \in F_m$ that have an equivalent in all source models are common in all source models and will finally be transferred to the reference model:

$$m\_n(n_m) = \begin{cases} 1 & \text{if } |Set(s\_n(n_m))| = |S| \\ 1 & \text{if } n_m \text{ is marked during steps 2 and 3} \\ 0 & \text{else} \end{cases}$$

$$m\_e(f_m) = \begin{cases} 1 & \text{if } |Set(s\_e(f_m))| = |S| \\ 1 & \text{if } f_m \text{ is marked during steps 2 and 3} \\ 0 & \text{else} \end{cases}$$

3.4. Step 2: Grouping

To group model elements, RefPA provides query constructs similar to those in SQL: `GROUP BY`, `CONTAINING`, and `WHERE`. The user applies `GROUP BY` to specify the property that constitutes a group and uses `CONTAINING` and `WHERE` to specify conditions that *all* activities (`WHERE`) or a *single* activity (`CONTAINING`) of a group must meet.

3.4.1. GROUP BY

`GROUP BY` is used to specify the activities' property for the grouping—that is, which property, such as annotated IT systems or executing role, must be equal for all of a group's activities. For instance, all of a source model's activities that process the same data object are grouped, and if two data objects are processed in each source model, there are two groups per model (e.g., I.1 for the first data object and I.2 for the second data object in source model I in Fig. 1). All activities of a group share common values for the property mentioned in the `GROUP BY` clause. However, `GROUP BY` cannot be used to ensure certain values for the property of the groups' activities (e.g., the data object must be "Application"), as it requires only that activities have equal values for the property mentioned in the clause without limiting the equal values.

Formally, the function *GroupBy* receives the property p_u selected by the user as input and returns the set of all groups, which is denoted as G. It uses the function *GroupPerValueInModel* to form the groups. This function creates one group gr with all activities of a model m_i that share a value v for the selected property. Each group is characterized by its set of activities A_{gr}, the activities' model, and the value that is common to all activities' selected property.

$GroupPerValueInModel\,(m_i, p_u, v) = \{gr\}$ with $gr = (v, m_i, A_{gr})$, $A_{gr} = \{a \in A_i | v \in props_i\,(a, p_u)\}$ and $A_{gr} \neq \varnothing$

The groups of each individual model are consolidated by the function *GroupsPerModel*:

$$GroupsPerModel\,(m_i, p_u) = \{GroupPerValueInModel\,(m_i, p_u, v)\,|\,v \in Val \in V_i\}$$

The function *GroupsPerModel* returns all groups of one model. Since *GroupsPerModel* iterates through all values that occur, it

identifies all possible groups for the property. Since the property values are sets, v must be contained in the activities' sets – denoted as *Val* – for the selected property. For instance, if an activity processes more than one data object and one of these data objects is v, the activity belongs to the group of v.

The function *GroupBy* returns the set of all groups of model elements. For this purpose, it consolidates the groups from all source models:

$$G := GroupBy\,(p_u) = \{GroupsPerModel\,(m_i, p_u)\,|m_i \in S\}$$

We present the syntax to construct RefPA queries using the Extended Backus–Naur form [61]. At the top, we have a `queryStatement`, which can be subdivided using various constructs. A `groupBy` clause is obligatory in a RefPA query since the aim is to form and evaluate groups. At this stage, a `queryStatement` consists only of a `groupBy` clause:

```
queryStatement = groupBy;
```

A `groupBy` clause starts with the keyword `GROUP BY`, which is followed by the indication of the property `p` that is used to group activities. The list of properties varies from modeling language to modeling language. Since `p` depends on the modeling language that was used to create the source models, we exemplarily mention the properties listed in Section 3.2 in our specification of `p`.

```
groupBy = "GROUP BY" p;
p = "Caption" | "Data Object" | "IT System" | "Role" | "External
    Stakeholder" | "Legal Regulation" | "Cost" | "Time" | "Probabil-
    ity";
```

The formalization is illustrated with the two exemplary BPMN process models in Fig. 2. We assume that the following `GROUP BY` clause is applied to the example:

`GROUP BY` *Data Object*

The `GROUP BY` clause groups the activities according to the property Data Object. The function *GroupPerValueInModel* creates two groups for model A and one group for model B. The first group of model A contains the activities `A1`, `A2`, `A3`, and `A4`, which share the value Data Object 1 for the property Data Object, and the second group of model A consists of the activity `A4` since it has the value Data Object 2. Hence, the activity `A4` belongs to two groups. All of model B's activities have the value Data Object 1 in common, which leads to one group with `B1`, `B2`, `B3` and `B4`. The groups are then brought together per model by *GroupsPerModel*. The set of all three groups is finally returned by the function *GroupBy*.

3.4.2. CONTAINING

While `GROUP BY` specifies the grouping property, `CONTAINING` and `WHERE` define conditions that groups must fulfill. `CONTAINING` is used to define conditions that must be valid for at least *one* activity of a group, so it ensures that a group contains at least one activity with certain property values. For instance, it can specify that each group must consist of at least one activity that is processed by a certain role. To define conditions for multiple activities, several `CONTAINING` statements can be included in a query, each of which must be met by one activity of a group. The statements can be fulfilled by different activities but they can also be fulfilled by the same activity.

Formally, the function *Containing* receives the set of groups G returned by the function *GroupBy* and a statement sc_u created by the user as input. It returns the set of all groups that fulfill the conditions of the statement.

sc_u can be an *atomic statement* or a *combined statement*. To specify the conditions, the user defines an atomic statement *asc*, which consists of three parameters: A property p_u, a value v_u, and an operator oac_u, the last of which denotes the relationship between value and property. The user can combine atomic statements to represent more complex conditions. A combined statement *csc* consists of two parameters: a set of statements ssc_u and an operator occ_u, which denotes the relationship between the statements. The two types of statements are formalized as:

- $asc = (p_u, oac_u, v_u)$ with $oac_u \in \{>, \geq, <, \leq, =, \neq\}$
- $csc = (ssc_u, occ_u)$ with $occ_u \in \{AND, OR, XOR\}$ and ssc_u as a set of sub-statements. A sub-statement can be an atomic statement or another combined statement.

The function *EvalStateCont* checks whether a group *gr* fulfills a statement sc_u:

$$EvalStateCont\,(gr, sc_u) = \begin{cases} 1 & if (sc_u\ is\ csc) \wedge (occ_u = AND) \\ & \wedge((\prod_{sub \in ssc_u} EvalStateCont\,(gr, sub)) = 1) \\ 1 & if (sc_u\ is\ csc) \wedge (occ_u = OR) \\ & \wedge((\sum_{sub \in ssc_u} EvalStateCont\,(gr, sub)) \geq 1) \\ 1 & if (sc_u\ is\ csc) \wedge (occ_u = XOR) \\ & \wedge((\sum_{sub \in ssc_u} EvalStateCont\,(gr, sub)) = 1) \\ 1 & if\ (sc_u\ is\ asc) \wedge (oac_u = >) \\ & \wedge(\exists a \in A_{gr}\ with\ (v \in props_i\,(a, p_u)) \wedge (v > v_u)) \\ 1 & if\ (sc_u\ is\ asc) \wedge (oac_u = \geq) \\ & \wedge(\exists a \in A_{gr}\ with\ (v \in props_i\,(a, p_u)) \wedge (v \geq v_u)) \\ 1 & if\ (sc_u\ is\ asc) \wedge (oac_u = <) \\ & \wedge(\exists a \in A_{gr}\ with\ (v \in props_i\,(a, p_u)) \wedge (v < v_u)) \\ 1 & if\ (sc_u\ is\ asc) \wedge (oac_u = \leq) \\ & \wedge(\exists a \in A_{gr}\ with\ (v \in props_i\,(a, p_u)) \wedge (v \leq v_u)) \\ 1 & if\ (sc_u\ is\ asc) \wedge (oac_u = ==) \\ & \wedge(\exists a \in A_{gr}\ with\ (v \in props_i\,(a, p_u)) \wedge (v = v_u)) \\ 1 & if\ (sc_u\ is\ asc) \wedge (oac_u = \neq) \\ & \wedge(\exists a \in A_{gr}\ with\ (v \in props_i\,(a, p_u)) \wedge (v \neq v_u)) \\ 0 & else \end{cases}$$

An atomic statement is valid if the value inputted to the function and the value for the inputted property of one activity of the group have the relationship that is specified by the operator (lines 4 to 9). For example, line 4 checks whether the statement is an atomic statement, the operator is $>$, and there is an activity in the group whose value for the property is greater than the value specified by the user. A combined statement is valid if either all sub-statements *sub* are valid (AND), at least one sub-statement is valid (OR), or exactly one sub-statement is valid (XOR) (lines 1 to 3). To check the validity of each sub-statement, the function calls itself for each sub-statement. For example, line 1 checks whether the statement is a combined statement, the operator is *AND*, and the product of *EvalStateCont* for all sub-statements equals 1; that is, it checks that all sub-statements are valid. In all other cases, a statement is not valid (line 10).

The function *Containing* returns the set of all groups that fulfill the user's statement; that is, the function *EvalStateCont* returns

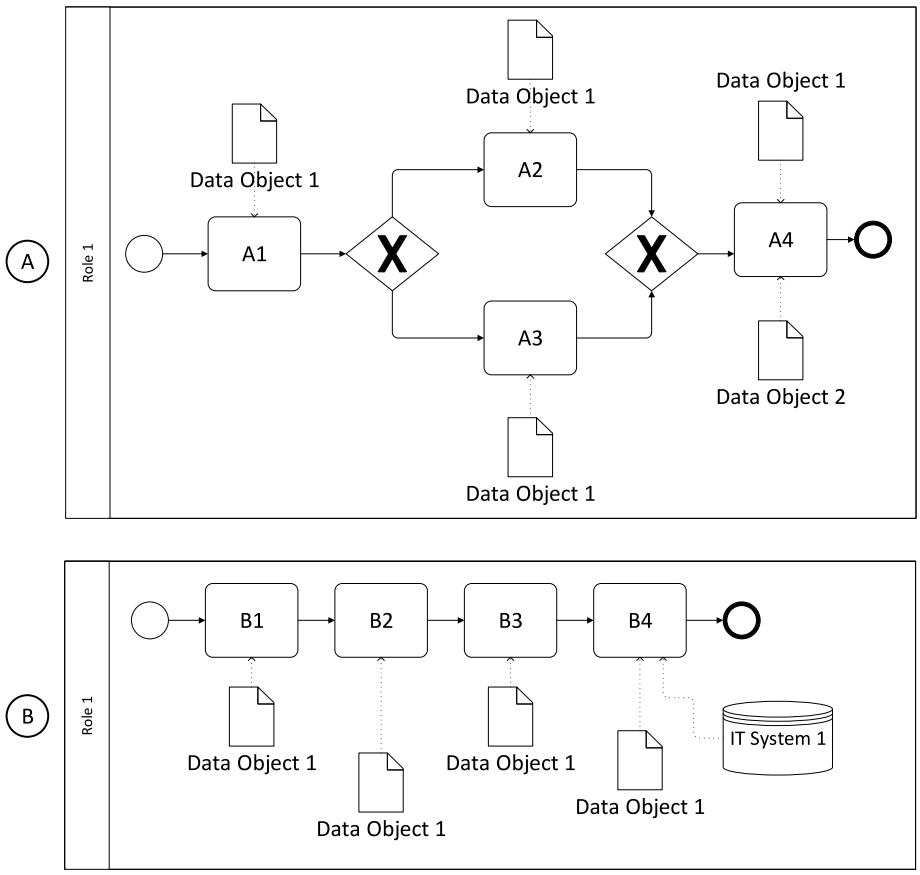

Fig. 2. Two exemplary process models, A and B.

the positive result 1 for such groups; and the set G is updated such that:

$G := Containing\ (G, sc_u) = \{gr \in G | EvalStateCont\ (gr, sc_u) = 1\}$

A RefPA query can contain several `containing` clauses, which can be fulfilled by the same or different activities of a group:

```
queryStatement = groupBy {containing};
```

A `containing` clause begins with the keyword `CONTAINING`. The user can then specify an atomic statement `asc` or combined statement `csc`. In an atomic statement, the user must specify the property `p`, an operator `oac`, and a value `string`. We do not detail `string`, which can be any arbitrary sequence of characters. A combined statement consists of at least two sub-statements and a combining operator but can also contain an arbitrary number of additional sub-statements and combining operators.

```
containing = "CONTAINING" sc;
sc = asc | csc;
asc = "(" p oac string ")";
oac = ">" | "≥" | "<" | "≤" | "=" | "≠";
csc = "(" sc occ sc {occ sc} ")";
occ = "AND" | "OR" | "XOR";
```

We assume that the following clauses are applied to the example in Fig. 2:

```
GROUP BY Role
CONTAINING (((Data Object = Data Object 1) OR (Data Object = Data Object 3))
      AND (Role = Role 1))
CONTAINING ((IT System = IT System 1) OR (IT System = IT System 2))
```

According to the `GROUP BY` clause, the activities are grouped according to their roles—that is, we have one group for source model A and one group for source model B. Each of the two `CONTAINING` clauses must be fulfilled by one activity of each group or the group will not be considered for further filtering and evaluation. In each group, there must be an activity that processes Data Object 1 or Data Object 3 and is executed by Role 1. Since this condition is valid for all activities, both groups fulfill this condition. The second `CONTAINING` clause requires one activity with IT System 1 or IT System 2. Since there is no such activity in the group of model A, this group is eliminated. The only remaining group is model B's group, as it includes activity B4, which is supported by IT System 1.

3.4.3. WHERE

`WHERE` is applied to filter the set of groups according to their activities' properties. Certain values and conditions for activities can be specified that must be valid for *all* of a group's activities; for example, all activities of a group are not allowed to be processed by a certain role.

The formalization of *Where* is the same as the formalization of *Containing*, with one exception: $\exists$ is substituted by $\forall$ in lines 4 to 9 in the specification of *EvalStateCont* since the conditions must be valid for *all* of a group's activities, not just one.

The user can also integrate a `where` clause into a RefPA query:

```
queryStatement = groupBy {containing} [where];
```

A `where` clause is similar to a `containing` clause, as the only difference is the keyword at the beginning of each clause.

```
where = "WHERE" sc;
```

We apply the following clauses to the example in Fig. 2:

```
GROUP BY Role
WHERE (((Data Object = Data Object 1) OR (Data Object = Data Object 3))
    AND (Role = Role 1))
```

The groups are created according to roles, so there is one group per model. All of the groups' activities must process either Data Object 1 or Data Object 3 and must be performed by Role 1. Since all activities are associated with Data Object 1 and executed by Role 1, both groups fulfill this condition.

3.5. Step 3: Evaluation

RefPA provides two query constructs with which to evaluate groups: `ORDER BY` and `HAVING`. The user applies `ORDER BY` to specify how the groups are ordered and applies `HAVING` to filter the set of groups according to certain aspects of evaluation.

RefPA provides a set of ranking criteria to facilitate evaluation of groups with these constructs. We observed the ranking criteria in the literature on best practices and developed additional ranking criteria based on the properties provided by process modeling languages, as listed in Section 3.2. Table 3 shows the ranking criteria, not all of which are relevant to every scenario; in some scenarios, public administrations aim for high values for a particular criterion and low values for the same criterion in other scenarios. Therefore, the user must select appropriate criteria based on the scenario's goals and circumstances and the user's preferences. To compute the results of a query, we developed calculations for the ranking criteria, shown in Appendix B.

3.5.1. HAVING

The query construct `HAVING` uses these ranking criteria and can be applied to define conditions that must be fulfilled by the entire group as an entity. For example, each group must have five media changes at most or be a connected subgraph when it will be considered for further processing.

Formally, and similar to *Containing*, the function *Having* receives the current set of groups G returned by the function *Where* and a statement sh_u that the user creates as input. It returns the set of all groups that fulfill the requirements of the statement.

The function *Having* processes two types of statements:

- $ash = (c_u, oah_u, v_u)$ with $oah_u \in \{>, \geq, <, \leq, =, \neq\}$
- $csh = (ssh_u)$ with ssh_u as a set of sub-statements. A sub-statement can be an atomic statement or another combined statement

An atomic statement is a three-tuple with a ranking criterion, an operator oah_u, and a value v_u. $c_u: G \rightarrow \mathbb{R}$ is the function of a ranking criterion (cf. Appendix B). A combined statement consists of a set of statements and no operator since the statements are always related by an AND connector.

The calculation of $EvalStateHav$ follows the calculation of $EvalStateCont$:

$$EvalStateHav\,(gr, sh_u) = \begin{cases} 1 & if (sh_u \text{ is } csh) \wedge ((\prod_{sub \in ssh_u} EvalStateHav\,(gr, sub)) = 1) \\ 1 & if\ (sh_u \text{ is } ash) \wedge (oah_u = >) \wedge (c_u(gr) > v_u) \\ 1 & if\ (sh_u \text{ is } ash) \wedge (oah_u = \geq) \wedge (c_u(gr) \geq v_u) \\ 1 & if\ (sh_u \text{ is } ash) \wedge (oah_u = <) \wedge (c_u(gr) < v_u) \\ 1 & if\ (sh_u \text{ is } ash) \wedge (oah_u = \leq) \wedge (c_u(gr) \leq v_u) \\ 1 & if\ (sh_u \text{ is } ash) \wedge (oah_u = =) \wedge (c_u(gr) = v_u) \\ 1 & if\ (sh_u \text{ is } ash) \wedge (oah_u = \neq) \wedge (c_u(gr) \neq v_u) \\ 0 & else \end{cases}$$

For example, the second line checks whether the statement is an atomic statement, the operator is $>$, and the value of the ranking criterion's function for the group is greater than the value specified by the user.

The function *Having* returns the set of all groups that fulfill the requirements of the statement and updates the set G:

$$G := Having\,(G, sh_u) = \{gr \in G | EvalStateHav\,(gr, sh_u) = 1\}$$

A RefPA query can contain a `having` clause:

```
queryStatement = groupBy {containing} [where] [having];
```

A `having` clause begins with the keyword `HAVING`, which is followed by an atomic statement `ash` or combined statement `csh`. An atomic statement consists of a criterion `crit`, an operator `oah`, and a value `string`. The terminal expressions for `crit` result from the criteria introduced in Section 3.5. A combined statement can consist of several sub-statements but must have at least two.

```
having = "HAVING" sh;
sh = ash | csh;
ash = "(" crit oah string ")";
crit = "Documents" | "Change of Documents" | "Organizational Units"
    | "Change of Organizational Units" | "Software Systems" |
    "Changes of Software Systems" | "IT Support" | "External Con-
    tacts" | "External Participants"| "Legal Foundations" | "Legally
    Unnecessary Steps" | "Media Changes" | "Electronic Processing" |
    "Paper-based Processing" | "Cost" | "Lead Time" | "Processing
    Steps" | "Absolute Frequency"| "Connectivity";
oah = ">" | "≥" | "<" | "≤" | "=" | "≠";
csh = "(" sh "AND" sh {"AND" sh} ")";
```

To illustrate the `HAVING` construct, we apply the following clauses to the example in Fig. 2:

```
GROUP BY Role
HAVING ((Processing Steps < 5)
        AND (Organizational Units ≤ 2)
        AND (IT Support ≥ 1))
```

Again, the groups are formed according to the activities' roles. Both groups consist of four activities (fewer than five processing steps), and have one role (fewer than two organizational units). Thus, both groups fulfill the first two atomic statements. The group of model B has one activity with an annotated IT system, so this group also fulfills the third atomic statement. However, the group of model A has no activity with an IT system, so this group does not fulfill the third atomic statement and is eliminated from further processing.

3.5.2. ORDER BY

The user applies `ORDER BY` to specify the ranking criteria to be used to rate the groups. The user indicates for each criterion whether groups are ordered in an ascending or descending order since in some cases high values for a criterion are desirable and in other cases low values are preferable (e.g., criterion 1).

Formally, the function *OrderBy* receives the final set of groups G returned by the function *Having* and a selection of ranking criteria $crit_u$ that the user specifies as input, and returns a set of preordered sets of G. *OrderBy* creates a preordered set for each value of the grouping property; that is, it orders the groups separately for each value that occurs in the property value sets. For instance, if there are two data objects A and B, then *OrderBy* creates two ordered sets, one for the groups whose activities all deal with data object A and one for the groups whose activities all deal with data object B. The result of *OrderBy* is the final result of a RefPA query.

Table 3
Overview of ranking criteria.

#	Criterion	Rationale
1	Documents Source: [62]	Documents are inputs and outputs of services in public administrations that flow through their processes [63]. Services like the tax declaration are characterized by many forms and attachments, so this criterion counts the number of data objects that are processed by a group's activities. A higher number of documents can be negative since more elements must be managed, increasing complexity, but a higher number of documents can also be positive since documents can indicate a detailed structure.
2	Change of documents Source: [59]	Because of the importance of documents for public administrations, this criterion minimizes the changes in processed documents, so it counts changes in data objects. First processing document D, then E, and finally D again might not be a good solution since retrieving documents requires effort. This criterion is useful only if a group consists of a sequence of connected activities and does not contain a set of individual activities that are widely spread across a model.
3	Organizational units Source: [64]	Analogous to criterion 1, this criterion counts the number of roles of a group's activities. A high value can be negative since many organizational units must be coordinated, but it can also be positive since the expertise of many organizational units can be complementary. This aspect must be reflected based on the context, as there are various complex services, such as those in the building permission area or the plan approval procedure, where complementary expertise is important.
4	Change of organizational units Source: [64]	Analogous to criterion 2, this criterion computes the number of changes with regard to the roles of a group's activities. A low value is beneficial since frequent exchanges between organizational units mean that multiple people must be familiar with a case initially or repeatedly, which constitutes avoidable effort. Exchanges and hand-overs between organizational units do not create value for citizens and companies.
5	Software systems Source: [65]	Analogous to criterion 1, this criterion counts the number of IT systems in a group's activities. A high value can be negative since many software systems must be coordinated and interfaces must be maintained, but it can also be positive since the impact of an attack or breakdown can be limited. If a public administration uses a single software system and this system fails, then the public administration is incapacitated.
6	Changes of software systems Source: [60]	Analogous to criterion 2, this criterion computes the number of changes with regard to the IT systems used in a group's activities. A change in software systems indicates that one system hands over a case to another system, so a low value for this criterion is desirable since exchanges and hand-overs between internal software systems do not create value for citizens and companies.
7	IT support Source: [64]	Since public administrations are encouraged to provide services electronically [66], this criterion counts how many of a group's activities have at least one assigned IT system. If this criterion has a high value, then many activities are supported by at least one software system.
8	External contacts Source: [67]	Since laws define public administrations' service portfolios, citizens can be forced to engage with public administrations, leading to displeasure when citizens engage with them [68]. To minimize the involvement of external participants like citizens and decrease citizens' discomfort, this criterion counts how many activities involve an external stakeholder, which is analogous to criterion 7.
9	External participants Source: [59]	Analogous to criterion 1, this criterion counts the number of external stakeholders of a group's activities. Many services in public administrations, such as plan-approval procedures, have many participants. Therefore, a higher value can be negative since many external participants must be coordinated, but it can also be positive since the expertise of many external participants complements the public administration's expertise.
10	Legal foundations Source: [58]	The services of public administrations are defined and restricted by legal foundations [20]. Performing as many legally necessary activities as possible can be a quality criterion since activities that have no legal foundation could be avoidable. For this purpose, this criterion counts how many activities have at least one assigned legal regulation, which is analogous to criterion 7.
11	Legally unnecessary steps Source: [58]	Since performing as many legally necessary activities as possible can be a quality criterion, this criterion counts how many activities have no legal regulation, which is analogous to criterion 7. However, additional steps might enhance service satisfaction, such as by informing citizens about the status of a government service instance.
12	Media changes Source: [69]	A media change is a typical weakness of a process [70]. A low value for this criterion is desirable since changes between media, such as a transfer from paper to an electronic medium, rarely create value for citizens and companies and should be avoided. This criterion counts how many changes between paper and electronic processing occur, which is analogous to criterion 7.
13	Electronic processing Source: [58]	Public administrations are encouraged to provide services electronically [66], as doing so increases internal efficiency. This criterion counts how many activities process a data object using an IT system.
14	Paper-based processing Source: [58]	Since public administrations are encouraged to provide services electronically, this criterion determines the degree of non-digitization by counting how many activities process a data object without an IT system.
15	Cost Source: [71]	Since public administrations should use their resources efficiently [72], this criterion adds up the cost of a group's activities.
16	Lead time Source: [73]	As public administrations should use their resources efficiently, this criterion adds up the lead times of a group's activities.
17	Processing steps Source: [60]	Since a small number of activities may indicate a short process, this criterion adds the number of activities in a group.
18	Absolute frequency Source: [34]	Since best practices can be practices that have been approved by many organizations, this criterion detects common practices. For each of a group's activities, it calculates the frequency of occurrences in the source models and returns the average frequency value.
19	Connectivity Source: [74]	In general, groups do not have to be connected, but criteria such as criterion 2 are meaningful only if a group is a connected subgraph and is not widely spread across the model. In addition, the final reference model must be a connected graph. Therefore, this criterion determines whether the group is a connected subgraph or not.

The function *Order* creates a preordered set for groups based on the user's preferences specified in $crit_u$. $crit_u = ((c_1, o_1), \ldots, (c_l, o_l), \ldots, (c_n, o_n))$ represents the selection of ranking criteria by the user: $c_l: G \rightarrow \mathbb{R}$ is the function of a ranking criterion (cf. Appendix B), and $o_l \in \{0, 1\}$ indicates whether the groups are to be ordered ascendingly (1) or descendingly (0) according to the criterion. Since groups may need to be ordered descendingly but we always sort ascendingly, we adapt $crit_u$ as $crit_u^*$.

For this purpose, we define an adapted function of a ranking criterion $c_l^* : G \times \{0, 1\} \rightarrow \mathbb{R}$ as follows:

$$c_l^*(gr, o_l) = \begin{cases} c_l(gr) & \text{if } o_l = 1 \\ -c_l(gr) & \text{if } o_l = 0 \end{cases}$$

This function changes the groups' values for the ranking criteria if we need to sort descendingly. In this case, we multiply all values by -1 so all groups can be ordered ascendingly.

$Order\left(G, crit_u^*\right) = (G, \lesssim)$ with $crit_u^* = (c_1^*, c_2^*, \ldots, c_n^*)$ orders lexicographically and returns a preordered set with a total preorder $\lesssim$ for G as follows [75]:

$(c_1^*(gr_1, o_1), c_2^*(gr_1, o_2), \ldots, c_n^*(gr_1, o_n))$	gr_1 precedes gr_2 in the preordered set if
$\lesssim (c_1^*(gr_2, o_1), c_2^*(gr_2, o_2), \ldots, c_n^*(gr_2, o_n))$	
$:\Leftrightarrow (c_1^*(gr_1, o_1), c_2^*(gr_1, o_2), \ldots, c_n^*(gr_1, o_n))$	their values for all ranking criteria are equal,
$= (c_1^*(gr_2, o_1), c_2^*(gr_2, o_2), \ldots, c_n^*(gr_2, o_n))$	
or $(c_1^*(gr_1, o_1) < c_1^*(gr_2, o_1))$	or gr_1 has a lower value for the first criterion,
or $(c_1^*(gr_1, o_1) = c_1^*(gr_2, o_1)$ and $c_2^*(gr_1, o_2) < c_2^*(gr_2, o_2))$	or the groups' first values are equal and gr_1 has a lower value for the second criterion,
…	…
or $(c_1^*(gr_1, o_1) = c_1^*(gr_2, o_1), \ldots, c_{n-1}^*(gr_1, o_{n-1}) = c_{n-1}^*(gr_2, o_{n-1})$ and $c_n^*(gr_1, o_n) < c_n^*(gr_2, o_n))$ with $gr_1, gr_2 \in G$.	or the groups' values are equal for all criteria except the last, where gr_1 has a lower value.

The function $Filter(G, v)$ filters the set of groups G by returning the set of all source models' groups that have the same value v for the grouping property in common; for instance, the groups' activities all process the data object (the grouping property) "Application" (the value): $Filter(G, v) = \{gr_1, gr_2, \ldots, gr_{|S|} \in G | v = v_1 = v_2 = \cdots = v_{|S|}\}$.

Finally, since there is a separate evaluation for the groups based on the value for the grouping property, *OrderBy* integrates preordered sets of *Order* into a single set:

$OrderBy\ (G, crit_u) = \{Order(Filter(G, v), crit_u^*)\}$ for all $v \in Val \in \bigcup V_i$, $1 \leq i \leq |S|$

An `orderBy` clause is obligatory in a RefPA query since the goal is to evaluate groups of model elements:

```
queryStatement = groupBy {containing} [where] [having] orderBy;
```

The keyword `ORDER BY` is the beginning of an `orderBy` clause. The user must mention at least one criterion and whether the groups are to be ordered ascendingly or descendingly according to this criterion. Additional criteria can be added.

```
orderBy = "ORDER BY" "(" crit o {"," crit o} ")";
o = "ASC" | "DESC";
```

The following clauses are applied to the example in Fig. 2:

```
GROUP BY Data Object
ORDER BY (Processing Steps ASC, Organizational Units ASC, IT Support DESC)
```

Since there are two data objects in the source models, the groups are separated into two parts. The first part consists of the groups that deal with Data Object 1 (one group in model A and one group in model B), and the second part consists of the group that deals with Data Object 2 (one group in model A). All groups are ranked ascendingly based on the processing steps, then ranked ascendingly based on the organizational units, and finally ranked descendingly based on IT support. The second ranking is trivial since it consists of only one group, and the groups in the first ranking both have four activities and one pool, so the third criterion determines the order. The group of model B is ranked at the top since it has one activity with an IT system, and the group of model A has none.

3.5.3. Marking of groups

After submitting a RefPA query with the constructs explained above, the query is executed, and the resulting ranking is presented to the user. All groups that share the same value for the grouping property are ranked separately based on the ranking criteria. The user can select a group that is marked to finally be transferred to the reference model. In most cases, the group the user selects is the one at the top for each value of the grouping property since it represents best practices according to the ranking criteria. If a query result is empty, the user can specify a new query.

Formally, the user selects a group $gr = (v, m_i, A_{gr})$ from the results to be transferred to the reference model:

(1)	$\forall a_m \in A_m$ such that $\exists a_s \in (A_{gr} \cap Set(s\_n(a_m)))$: $m\_n(a_m) := 1$	All activities of the merged model that have an activity in their set of source nodes that belongs to the selected group are marked.
(2)	$\forall b_{m1} \in B_m$ such that $\exists (n_{m1}, b_{m1}), (b_{m1}, n_{m2}) \in F_m$	For all gateways that directly connect two certain nodes of the merged model
	and $\forall b_{m1}, \ldots, b_{mk} \in B_m$ such that $\exists (n_{m1}, b_{m1}), (b_{m1}, b_{m2}), \ldots, (b_{mk-1}, b_{mk}), (b_{mk}, n_{m2}) \in F_m$	For all gateways that connect two certain nodes via a sequence of gateways (k denotes the length of the sequence)
	with $(n_{s1} \in Set(s\_n(n_{m1}))) \wedge (n_{s2} \in Set(s\_n(n_{m2})))$ and	
	• $n_{s1}, n_{s2} \in A_{gr}$	These two nodes are either activities of the selected group,
	• or $(n_{s1} \in A_{gr}) \wedge (m\_n(n_{m2}) = 1)$	or the first node belongs to the group and the second node has already been marked,
	• or $(n_{s2} \in A_{gr}) \wedge (m\_n(n_{m1}) = 1)$:	or the second node belongs to the group and the first node has already been marked:
	$m\_n(b_{m1}) := 1$	Mark such gateways
	...	
	$m\_n(b_{mk}) := 1$	
(3)	$\forall f_m = (n_{m1}, n_{m2}) \in F_m$ with $(n_{s1} \in Set(s\_n(n_{m1}))) \wedge (n_{s2} \in Set(s\_n(n_{m2})))$ and	For all edges of the merged model, where
	• $(n_{s1} \in A_{gr}) \wedge (m\_n(n_{m2}) = 1)$	the first node belongs to the group and the second node has been marked,
	• or $(n_{s2} \in A_{gr}) \wedge (m\_n(n_{m1}) = 1)$	or the second node belongs to the group and the first node has been marked,
	• or $n_{m1}, n_{m2} \in \{b_{m1}, \ldots, b_{mk}\}$	or both nodes are part of a sequence of gateways as stated in 2),
	• or $((n_{m1} \in \{b_{m1}, \ldots, b_{mk}\}) \wedge (m\_n(n_{m2}) = 1)) \vee ((n_{m2} \in \{b_{m1}, \ldots, b_{mk}\}) \wedge (m\_n(n_{m1}) = 1))$:	or the edge connects a gateway from 2) to a marked node:
	$m\_e(f_m) := 1$	Mark such edges.

First, the merged activities of all activities of the selected group are marked. Second, gateways are marked that connect activities of the group to other activities of the group or to nodes that have been marked before. The connection can be realized by a single gateway or a sequence of gateways. Third, the equivalents in the merged model of edges that connect nodes of the group to each other or to already marked nodes or the equivalents in the merged model of edges that connect gateways related to the group to other nodes are marked.

3.6. Step 4: Assembling the reference model

In the course of the assembling the reference model m_r, all unmarked nodes and edges are removed from the merged model:
$m_r = (A_r, B_r, F_r, P_r, V_r, props_r)$ with
$A_r = \{a_m \in A_m | m\_n(a_m) = 1\}$,
$B_r = \{b_m \in B_m | m\_n(b_m) = 1\}$,
$F_r = \{f_m \in F_m | m\_e(f_m) = 1\}$,
$P_r = \bigcup P_i$ and
$V_r = \bigcup V_i$

For each merged activity a_m, the user selects an atomic activity $a_u \in Set(s\_n(a_m))$ from one of the source models m_i that provides the property values for the activity in the reference model:

$\forall p \in P_i$: $props_r(a_m, p) := props_i(a_u, p)$ with $a_u \in (Set(s\_n(a_m)) \cap A_i)$ and $a_m \in (A_m \cap A_r)$

By default, the atomic activity that occurs first in one of the selected groups provides the property values for a merged activity since this activity is part of a best practice. However, if an activity has an equivalent in all source models but is not part of any selected group, then the user must make a choice. We decided to not take the union of the property values of all source activities since doing so would not be meaningful for properties with text, such as captions, or quantitative attributes, such as time.

Finally, the user must ensure that the model is syntactically correct and a connected graph. For example, the user can remove trivial gateways with only one ingoing and one outgoing edge. For this purpose, the user can apply the functionalities of existing process modeling tools or mechanisms that have been proposed in the literature [18,34] and modify the reference model if necessary.

4. Demonstration

We demonstrate the applicability and functionality of the entire method using an example based on two artificial BPMN process models, as visualized in Fig. 3. Fig. 4 provides the initial merged model (step 0) that was created manually. If we do not mention a property in one of the source model's activities, then the property's value for this activity is null. In the merged model, the sets of referenced source elements are mentioned within activity nodes and to the right of edges. For instance, the first node of the merged model refers to the atomic nodes `An1` and `Bn1`. Since there are two source models and the merged node refers to two atomic nodes, RefPA automatically recognizes the first node of the merged model as a common element. Fig. 4 illustrates the common parts (step 1).

During steps 2 and 3, the user exemplarily submits the following query first:

```
GROUP BY Data Object
CONTAINING (Caption = Receive Application)
WHERE (Role = Case Officer)
HAVING (Media Changes ≤ 2)
ORDER BY (Lead Time ASC, Paper − based Processing ASC)
```

The activities are grouped according to their processed data objects in the `GROUP BY` clause since government service delivery is characterized by the processing of documents [76]. We assume that the goal is to increase process efficiency and improve the degree of digitization [1,77], so the `ORDER BY` clause orders the

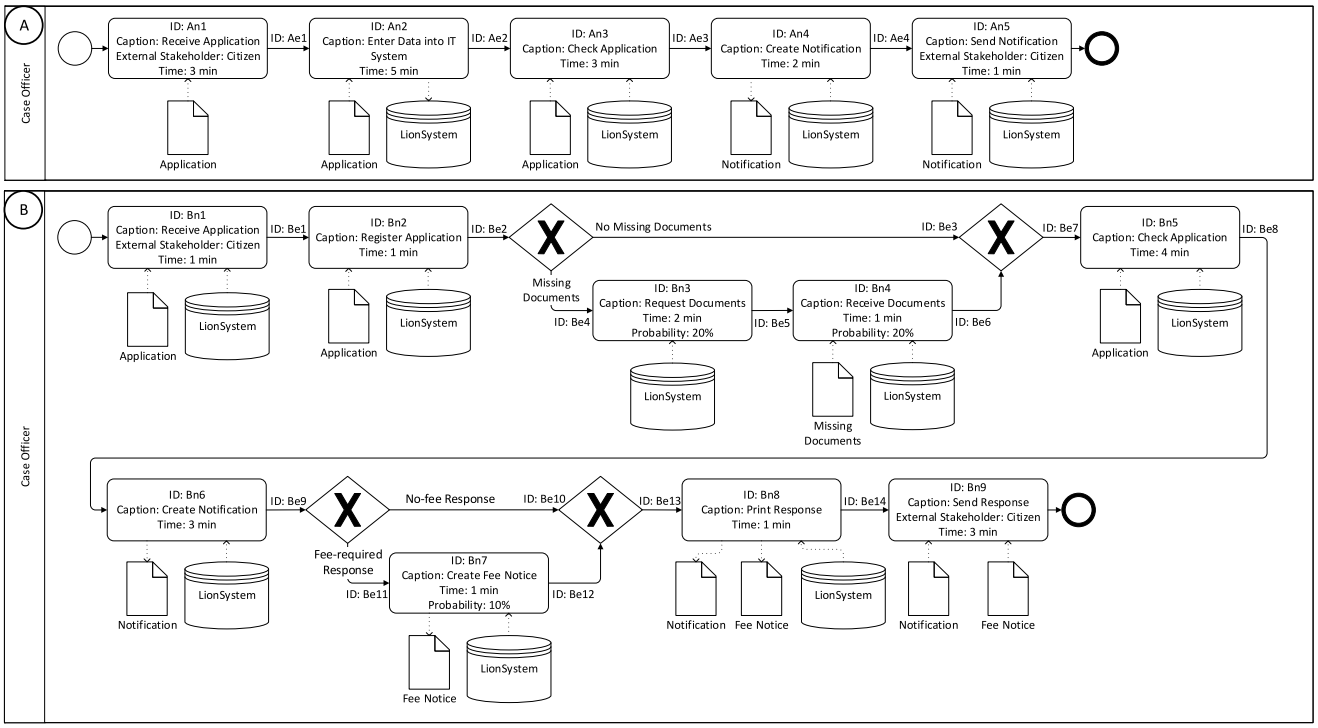

Fig. 3. Two exemplary process models A and B.

Table 4
Result of the first query.

Grouping		Evaluation	
Data object	Model	Lead time	Paper-based processing
Application	B	6 min	0
	A	11 min	1

Table 5
Result of the second query.

Grouping		Evaluation	
Data object	Model	Lead time	Paper-based processing
Application	B	6 min	0
	A	11 min	1
Fee notice	B	5 min	1
Missing documents	B	1 min	0
Notification	A	3 min	0
	B	7 min	1

groups according to lead time and paper-based processing (cf. Appendix B) to identify the fastest process segments and reduce the volume of paper documents in the process. The `HAVING` clause of this exemplary query limits the number of media changes so the process remains electronic as often as possible. The `WHERE` clause ensures that the same role processes all activities and does not have to hand work over to another role. As the first query is formulated to deal with the start of the process, we formulate an exemplary `CONTAINING` clause to ensure that at least one activity receives an application.

The automatically generated result of the query, shown in Table 4, is based on four data objects in the source models: "Application", "Missing Documents", "Notification", and "Fee Notice". Since the groups regarding "Missing Documents", "Notification", and "Fee Notice" do not contain an activity with the caption "Receive Application" and do not fulfill the `CONTAINING` clause, the only ranking is regarding "Application". The group of model B is superior to the group of model A, so the user selects this group and marks the node `Bn2` and the edge `Be1`. The merged model at this stage is visualized in Fig. 4.

To analyze the remaining parts of the models, the user removes the `CONTAINING` clause from the query since the user is now interested not only in the source models' start but in all parts of the models:

```
GROUP BY Data Object
WHERE (Role = Case Officer)
HAVING (Media Changes ≤ 2)
ORDER BY (Lead Time ASC, Paper − based Processing ASC)
```

The second query's result is presented in Table 5. The overview contains a ranking for the groups regarding all data objects processed in the source models. Since the user dealt with "Application" in the first query, s/he now focuses on "Notification". The group of model A is better than the group of model B since it is faster and processes fewer paper data objects. The user selects the group of model A regarding "Notification" in this example and marks the edge `Ae4`. The current merged model is visualized in Fig. 5.

An immediate rejection of an application that is due only to missing documents is not service-oriented, so the user decides that a public administration should inform the citizen about missing documents and give the citizen the opportunity to submit documents later. As processing missing documents is not relevant to all cases but only to exceptional cases, the user forms groups according to probability since the property probability indicates exceptional activities. This rationale leads to the following query:

```
GROUP BY Probability
ORDER BY (Lead Time ASC, Paper − based Processing ASC)
```

Table 6 displays the automatically computed result. An inspection of the source models reveals that all activities with a probability of 20 percent deal with the processing of missing

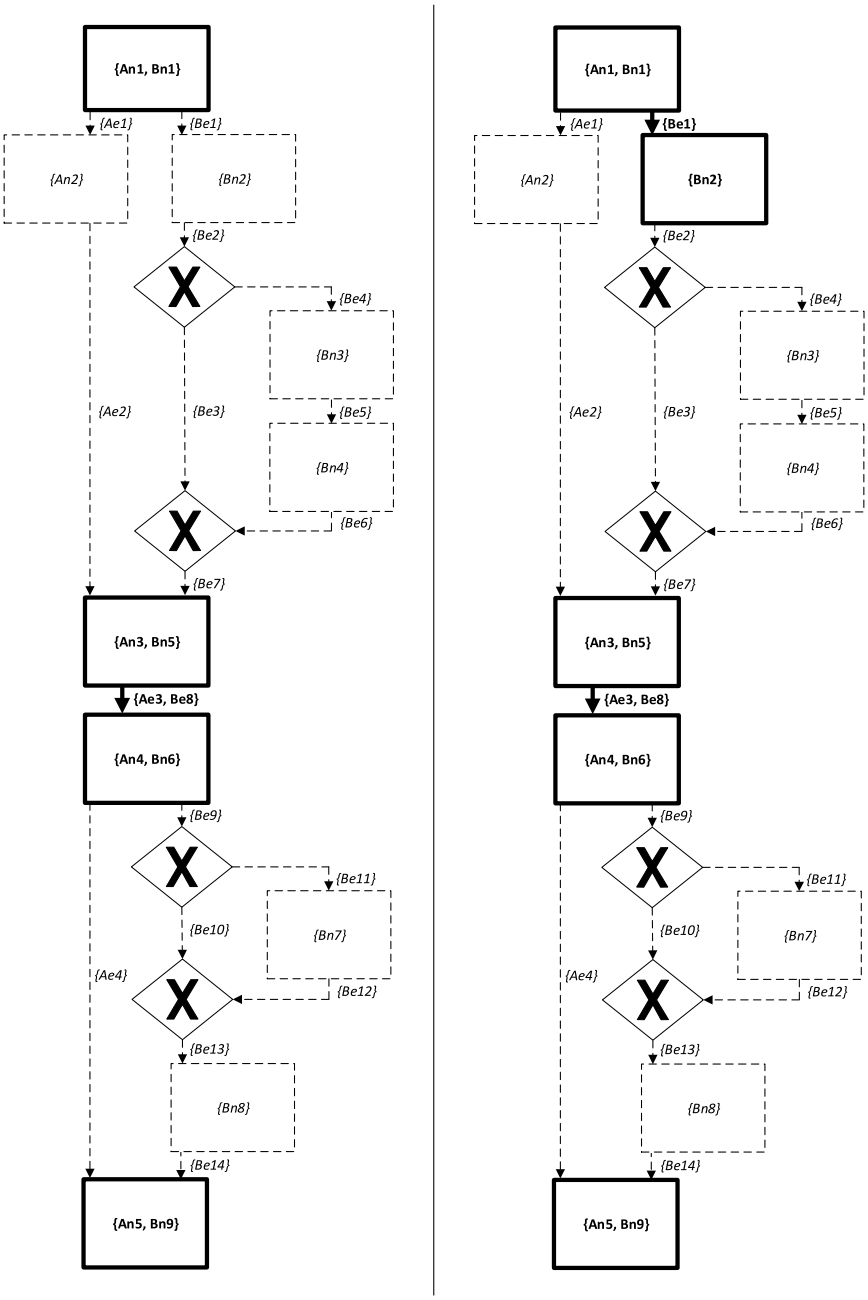

Fig. 4. The merged model initially (left) and after the user's first selection (right).

Table 6
Result of the third query.

Grouping		Evaluation	
Probability	Model	Lead time	Paper-based processing
10%	B	1 min	0
20%	B	3 min	0

documents. Since there is only one group for the probability 20 percent, the user selects this group of source model B and marks Bn3, Bn4, two gateways, and Be2, Be3, Be4, Be5, Be6, and Be7. The merged model at this stage is presented in Fig. 5.

The user does not formulate an additional query. The remaining data object, "Fee Notice", is not rated as relevant to the reference model since, in this example, the marked parts of source model A can include fee information in the notification. A separated data object for fee information is not necessary.

In the course of the assembling the reference model (step 4), the non-marked elements are automatically removed from the merged model (`An2`, `Bn7`, `Bn8`, `Ae1`, `Ae2`, `Be9`, `Be10`, `Be11`, `Be12`, `Be13`, `Be14`, two gateways). The user chooses the activities of the selected groups from steps 2 and 3 as providers of the reference model's activities' property values (`Bn1`, `Bn2`, `Bn3`, `Bn4`, `Bn5`, `An4`, `An5`). Since the reference model is syntactically correct and connected, the user performs no manual adaptations. Fig. 6 visualizes the resulting reference model.

5. Evaluation

This section reports on the evaluation of RefPA that we performed in a workshop using a software prototype. As a preliminary evaluation, we presented the main concepts at the European Conference on Information Systems 2016 and received valuable feedback that we used to refine RefPA's concepts and that we incorporated into the prototype.

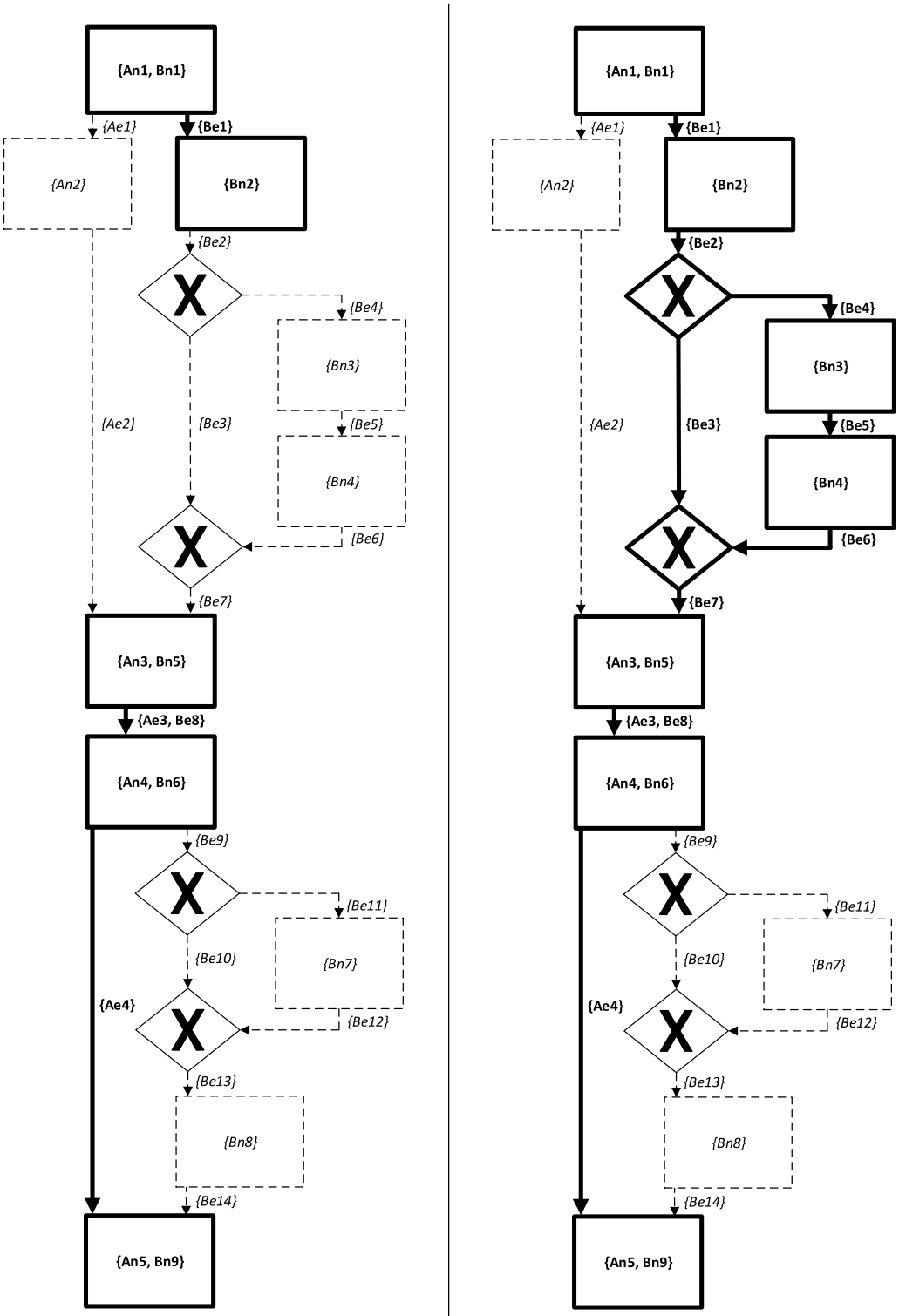

Fig. 5. The merged model after the user's second (left) and third (right) selections.

5.1. Implementation

We created a software prototype as a proof-of-concept and to evaluate RefPA. The focus of the implementation was on the algorithmic and procedural aspects of RefPA. The user interface was kept minimal to ensure that users could evaluate the method's usefulness, rather than the interface design. The RefPA prototype was implemented as a standalone Java desktop application with a Swing GUI. Choosing this powerful, well-understood, and popular programming language [78–80] was beneficial with regard to such aspects as platform independency, availability of libraries, flexibility, and availability of a mature development ecosystem.

After opening the prototype, the user is located in a view that allows him or her to import either "raw" process models or a previously assembled merged model. If only raw process models are presented, the tool has an integrated plug-in that enables the user to create the required merged model manually (step 0 in

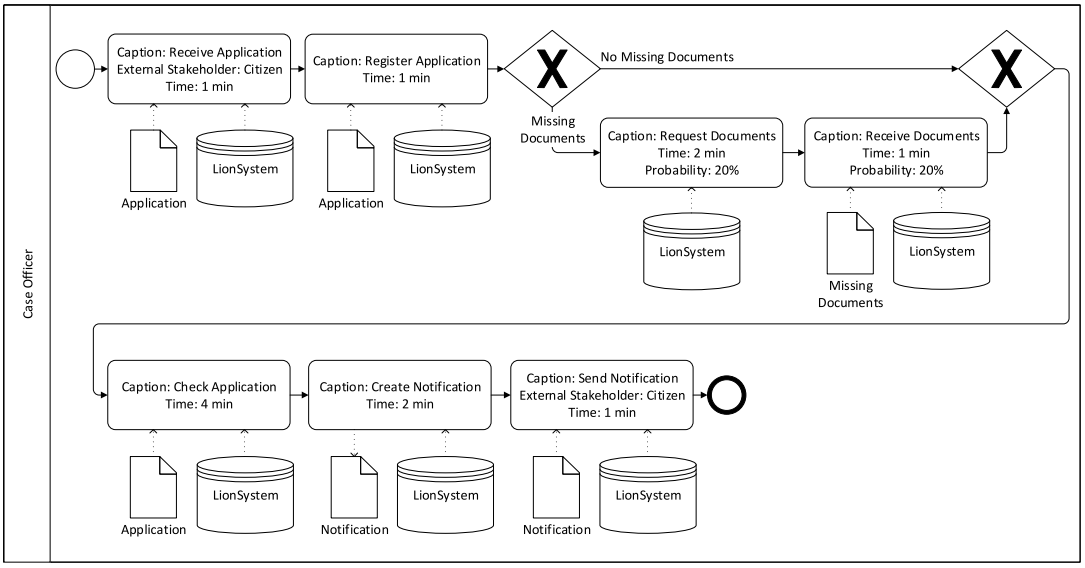

Fig. 6. Resulting reference model.

RefPA), supported by similarity values between the activities of the process models to be merged [81].

Once a merged model is loaded or created, the user can switch to the "Grouping and Evaluation" view where, consistent with RefPA's step 1, all common nodes in the merged model are automatically identified and marked in dark gray, while all common edges are displayed as solid arrows (see, e.g., bullet C in Fig. 8). All remaining elements are visualized in light gray or dotted arrows. Even though the RefPA query language in steps 2 and 3 is inspired by SQL, we decided against writing queries in the format suggested in Sections 3.4 and 3.5 (for examples, see Section 4). While the standard SQL syntax should be familiar to anyone with an IT background, it is likely to be difficult for people with other backgrounds. To ensure that these steps are accessible to all, RefPA queries that follow the syntax proposed in Sections 3.4 and 3.5 can be created using dropdowns and add/remove buttons (see Fig. 7). This approach is feasible since most of the non-terminals in the RefPA syntax are directly replaceable by elements from a fixed set of terminals, which are easy to represent as a dropdown list.

Fig. 7 shows an example of the second query stated in Section 4. In the first dropdown for `GROUP BY`, the user can select the property by which the activities should be grouped, which is "Data Object" in this example. Since the query uses no `CONTAINING` clause, no respective statement is added. However, for `WHERE`, a condition is added to restrict the result set to activities that are executed by a person whose "Role" is equal (=) to "Case Officer". Next, a `HAVING` statement is added with the "+" button. From the list of criteria, "Media Changes" is selected along with the operator "<=" and the manually added "2" as the target value. The retrieved results should be ordered according to "Lead Time" and "Paper-based Processing", both in ascending order, so "Lead Time" is selected from the dropdown of the first, mandatory `ORDER BY` statement. Via the "+", a second `ORDER BY` statement is added, where "Paper-based Processing" can be selected; "ASC" is the default order, so it does not need adjustment. Once the user is satisfied with the query, results can be obtained by clicking the "Show Results" button at the bottom of the query panel.

Query results are subsequently listed in the view, ordered by the specified ranking criteria (bullet A in Fig. 8). By clicking on a group in the overview, the affected nodes and edges are highlighted in orange in the merged model (bullet C). Based on the quantitative metrics and the visual impression, the user can

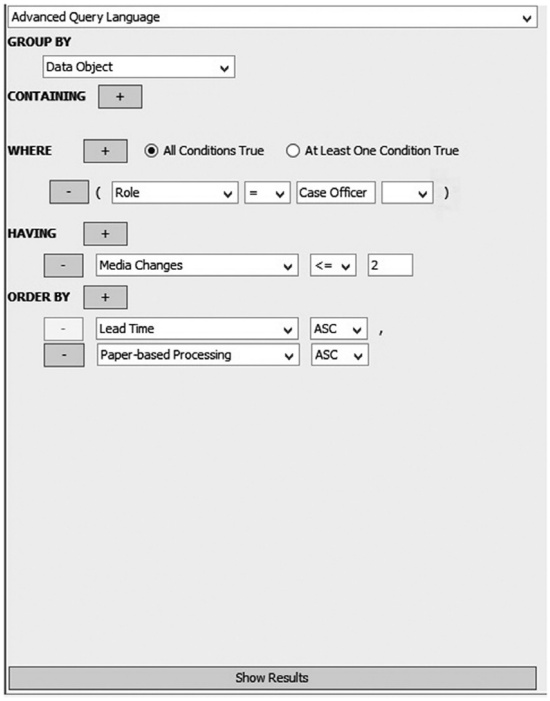

Fig. 7. Prototype—query editor for grouping and evaluation (steps 2 and 3).

decide which group represents the "best practice" to be included in the reference model. If more information about an activity is needed, the user can select an activity (bullet C) and look up the associated property values (bullet D). Since the merged model consists of merged activities, each of the merged activities' source activities can be selected via a dropdown for closer inspection (bullet D). If the user needs more information, s/he can access the source models by going back to the "Import" view. Once the user makes a decision, s/he adds the selected group by clicking the "Add Group to Reference Model" button (bullet B). Two other buttons can be used to switch between the merged model view (Fig. 8) and a visualization of the current version of the reference model that contains only marked elements (Fig. 9).

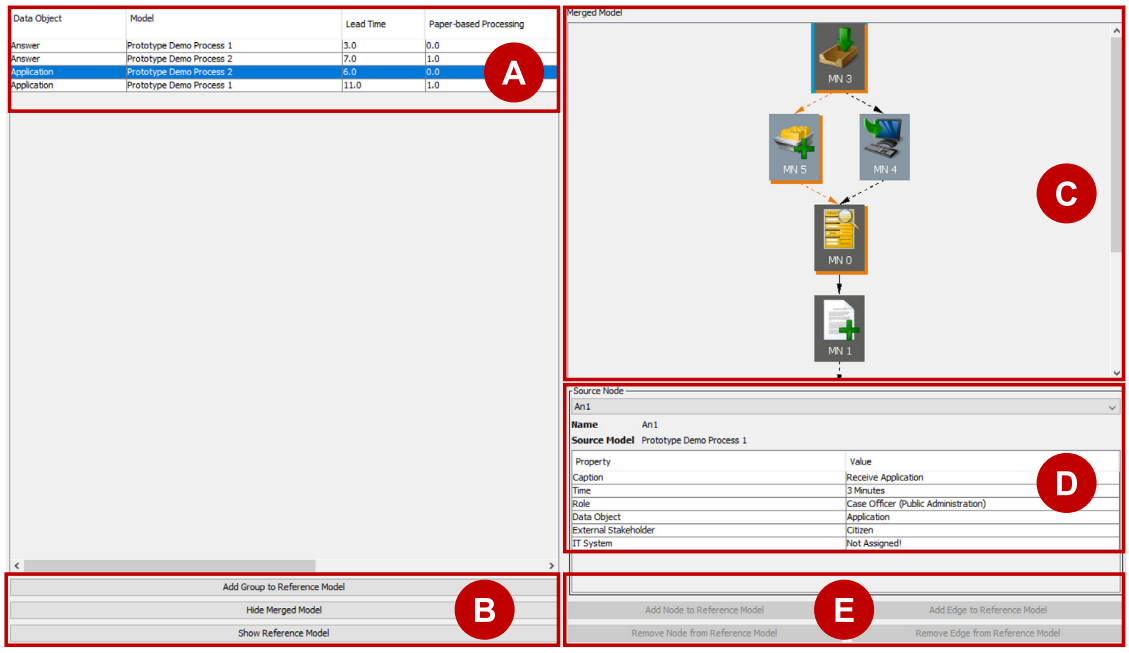

Fig. 8. Prototype—results of grouping and evaluation (steps 2 and 3).

Fig. 9. Prototype—reference model view: Unconnected (left) and connected (right) (step 4).

When the user presses the button "Show Reference Model" and the prototype presents the current reference model, the prototype warns the user if the model is not connected (step 4 in RefPA), as shown in Fig. 9. Here, the activity "Send Answer" (`Mn2`) has not yet been connected to the rest of the reference model. While unconnected items may be easily recognizable in demonstration cases like that presented here, this feature ensures that, even during the treatment of complex cases, the user can obtain a connected and usable process model. In the case presented here, the merged model offers two possible pathways to connect `Mn2` to the other nodes: one from "Create Answer" (`Mn1`) directly to "Send Answer" (`Mn2`) and one via "Print Answer" (`Mn6`). Once the user adds a group to establish one of those two options, the warning disappears and a connected reference model is presented (Fig. 9). For manual adaptions, the buttons in Fig. 8 (bullet E) allow the user to add or remove individual nodes and edges.

5.2. Workshop

5.2.1. Research design

To evaluate RefPA's usefulness, we applied the prototype in a workshop with nine domain and modeling experts. The participants' years of experience averaged 4.94 years in projects in public administrations and 5.17 years in projects covering conceptual modeling. We provided the participants with three real-world process models that represented the same process in three public administrations and contained 12, 15, and 24 activities. We performed a terminological standardization beforehand so the participants could focus on constructing the reference model. Two researchers supervised the workshop.

In the course of the workshop, we compared RefPA to a manual reference model construction but not to another semi-automatic inductive method since no similar method was available (cf. Section 2.2). All participants performed the manual and RefPA-based reference model construction so they could compare and evaluate the resulting reference models and methods themselves. We did not split the participants into two groups, with one group creating reference models manually and one group creating reference models with RefPA. Since a modeler creates a model for a certain purpose, the quality of a model is highly subjective. Therefore, it is not possible to compare the reference model that was created manually to the RefPA-based reference model and objectively decide which model's best practices are superior.

We divided the workshop into three parts: In the first part, the participants created a reference model manually on paper after receiving printed copies of the source models. Participants were given the goal of creating a generalized reference model for best practices and were not limited to a certain scenario.

After a short introduction to the functionality of RefPA and the use of the prototype, in the second part of the workshop each participant created a reference model with the RefPA implementation. We prepared the computers beforehand by installing RefPA and importing the terminologically standardized source models. The participants started by manually mapping similar activities to create the merged model and then used the query constructs and the query results to add elements to the reference model.

In the third part of the workshop, the participants completed a questionnaire that contained two closed-ended questions that asked for their experience, seven questions on a five-point Likert scale about the usefulness and applicability of the RefPA method and the query language, and three open-ended questions that dealt with prerequisites and asked for improvement suggestions. We provide the questionnaire in Appendix C.

5.2.2. Quantitative results

The results led to qualitative and quantitative insights. Table 7 presents an overview of the quantitative results regarding the questionnaire's questions on RefPA's usefulness and applicability. To account for the participants' experiences, we weighted their scores with their experiences when we calculated the means.

As Table 7 shows, the minimal weighted mean value of all statements is 3.15, and the maximum mean value is 4.27 (1 refers to "do not agree" and 5 refers to "fully agree"). The lowest mean values were in response to the statement that RefPA provides new and valuable suggestions and the statement that users can apply RefPA easily. The usefulness of the predefined steps received the highest mean value. Four statements received the minimum value of 1 and five statements received the maximum value of 5. The participants gave the highest minimum value of 3 to the usefulness of RefPA's predefined steps and to the likelihood of reusing RefPA by reference modelers. All statements received a maximum value of at least 4. The participants' responses led to a median of 4 for the majority of statements. The highest median is 4 and the lowest median is 2, where the latter belongs to RefPA's ease of use.

The results reveal that RefPA provides valuable support in constructing a reference model inductively. On average, all statements received more approval than rejection since all weighted mean values were above or equal to 3.15. Some negative outliers appeared, as indicated by their minimum values, but the median values reveal that those outliers had little impact on the participants' overall perceptions. In particular, the participants saw RefPA's predefined steps as providing valuable guidance to the construction process, as they perceived first detecting common parts in the source models with a merged model and then analyzing non-common parts with the query language to be a reasonable approach.

Although they rated RefPA as useful method, the participants saw more value for reference modelers in general than for process managers in public administrations in particular. Comparisons between RefPA's general applicability and its specific applicability in public administrations support this impression (questions #3 and #4, #6 and #7). The participants' qualitative feedback indicated that this view is due primarily to public administrations' limited progress in business process management by means of knowledge-sharing between organizations.

Although the semi-automatic inductive reference model construction is relevant to public administrations, it is far ahead of their current agendas. In general, though, the participants confirmed the usefulness of the RefPA method and agreed that public administrations would benefit from its application.

5.2.3. Qualitative results

A prerequisite for the successful application of RefPA is a software implementation with a high level of usability. The statement regarding RefPA's ease of use is the only statement of the questionnaire with a median that indicates a rejection from the participants, so in the participants' view, the usability of the current version of our prototype can be improved. In their answers to the open-ended questions of the questionnaire's third section, they contributed guidelines for an implementation of RefPA:

- An implementation should not only visualize the merged model after the user selects a group and marks the corresponding elements but should also illustrate potential consequences. The user should be aware of the effect of doing so before adding a group to the reference model, so the tool should be able to highlight potential groups in the merged model.
- Domain experts have difficulties understanding and using the query language, as it requires some prior IT knowledge. Although the current version of the prototype provides predefined query structures and does not force the user to define a query from scratch, it still relies on formal syntax. Since not only IT experts but also process managers in public administration should be able to use RefPA, an implementation should provide a graphic way to assemble queries that does not rely on formal specifications.
- In the current version, the user must understand the source models in detail (e.g., which properties are available to be referred in a query) and perform many manual steps. However, to make RefPA widely applicable, knowledge of the scenario should be sufficient. The tool should predefine typical default criteria and sequences of queries. It should ask the user questions to specify the reference model's scenario and goal and then suggest queries to the user that fit the scenario.
- To avoid requiring the user to know RefPA in detail and to accelerate the construction process, the tool should automatically add groups to the reference model that are ranked at the top instead of letting the user select one of the groups. Thus, the user would not have to inspect the groups that s/he could add to the reference model but would check a final reference model and adapt it if necessary (which must be done in step 4 anyway). This procedure sounds reasonable since the user expresses his or her preferences in the query, making subsequent selection of the best group from the overview unnecessary.
- The participants tended to take a node-centric view instead of a group-centric view, so they evaluated nodes individually in the manual construction, while RefPA compares

Table 7
Results of the questionnaire: RefPA's usefulness and applicability.

#	Statement	Weighted mean	Min	Max	Median
1	RefPA, with its criteria and query constructs, provided new and valuable suggestions for the resulting reference model.	3.15	1	4	3
2	RefPA's predefined steps gave a reasonable structure to the construction process.	4.27	3	5	4
3	RefPA is useful in constructing the reference model inductively.	4.03	2	5	4
4	RefPA is useful in public administrations' process management.	3.67	1	5	4
5	Users can apply RefPA easily.	3.18	1	4	2
6	As a reference modeler, I would use RefPA again.	4.04	3	5	4
7	As a process manager in a public administration, I would use RefPA again.	3.45	1	5	3

groups. In addition, the merged models in our examples became rapidly complex because of an increasing number of edges. For these two reasons, an implementation of RefPA could mark only nodes and no edges based on queries and let the user create the edges manually. A tool should have a modeling component that allows the user to modify the reference model.

Consequently, the evaluation revealed that RefPA is useful. RefPA's criteria and query constructs, in combination with the predefined steps, are suitable means with which to detect best practices in process models inductively and represent them in reference models. The participants indicated that RefPA is beneficial for reference modelers and process managers in public administrations. The prototype can be improved based on the participants' suggestions to increase its application.

6. Discussion and outlook

This paper contributes to research and practice an inductive method to develop semi-automatically reference process models that represent best practices. The method requires the integration of source models into a merged model and transfers selected elements from the merged model to the reference model (*FRQ3*). RefPA segments the source models by differentiating between common and non-common elements and providing three query constructs to form groups (*FRQ1*). It offers two SQL-like query constructs to evaluate such groups (*FRQ2*). RefPA is applicable to all graph-based process modeling languages (*FRQ4*). The main component of RefPA is a query language that allows users to form comparable segments in process models and evaluate these segments according to best-practice criteria. The list of process-related best-practice criteria in public administrations and the prototypical implementation of RefPA are contributions on their own.

Inductive methods usually return descriptive reference models [18], but the reference models that result from RefPA are both descriptive and prescriptive since they represent best practices from a limited set of process models. Such reference models are recommendations to align processes with those of the best organizations. However, since RefPA is based on a limited set of process models, better solutions may be available that are not included in the process models, so "best" is relative to the set of source models from which RefPA detects the best solutions. Since RefPA describes (descriptive) the best (prescriptive) available solutions in the source models, a user cannot assume that its suggestions are global best practices. In addition, the definition of best practices always depends on the user and the underlying goal: while some practices are best in some scenarios, they might be inferior to other practices in other scenarios, so the user should evaluate RefPA's results for best practices based on his or her knowledge and preferences. In short, RefPA returns suggestions for relative and perceived best practices.

Researchers can transfer RefPA's concepts to other perspectives of an organization, such as data models and organizational charts, but researchers must develop a different set of best-practice criteria for each type of model. For example, criteria for organizational charts could include the number of organizational units and the number of hierarchy levels. Transfer of RefPA to other sectors, such as retail or production companies, is promising and also requires adapting the ranking criteria. Since reference models can be theoretical contributions [10,82], researchers can apply RefPA to create a set of reference models that covers various functional areas of public administrations.

In practice, RefPA can facilitate knowledge-sharing between public administrations when practitioners apply RefPA in reference modeling. For example, an application scenario for a process manager is harmonizing the processes of subsidiaries to decrease time and cost. Several public administrations can apply RefPA to develop a reference process when they want to implement a common software system. Public administrations can also use RefPA in other kinds of scenarios, such as when a single public administration wants to introduce a new software system and must choose between two systems. It can construct two to-be process models, one with the first software system and one with the second and compare the two using RefPA to detect segments in which the first or the second system performs best. The segments that are most important can determine the decision.

RefPA is subject to some limitations and prerequisites. To apply RefPA, a user must fulfill the following prerequisites:

- The source models must be created with the same business process modeling language.
- The source models must have adequate information. RefPA uses property values to form and evaluate groups, so the source models must contain appropriate property values for their activities, such as roles and data objects. If only a few activities have values for some properties, process managers cannot apply RefPA meaningfully.
- As the source models come from different institutions, terminology may need to be standardized before RefPA is applied (e.g., [83–87]). For instance, different terms for property values cannot refer to the same entity (e.g., "invoice" and "bill").
- The semi-automatic comparison of process models requires common modeling conventions to be in place so modeling is performed consistently for all source models. For instance, if modeling of legal regulations is relevant to one source model, and the creator of another source model has not paid attention to modeling legal regulations, then these source models are not comparable (cf. criterion 10 in Table 3).
- An adequate method to create the inputted merged model must be selected. Users can perform this preprocessing step manually or rely on methods proposed in the literature, such as those for merging process models (e.g., [34,88–92]) and matching process models (e.g., [93–99]). The selected merging algorithm should rely on structural integration, not behavioral integration, since the latter does not necessarily reproduce edges that existed before.

- A user should have access to state-of-the-art process modeling tools that provide functionalities for the syntactic check and clean-up of process models or mechanisms proposed in the literature (e.g., [18,34]) to ensure that the result is a proper process model.

Ensuring that all prerequisites are met may be time consuming, so users must decide in each case whether it is worth the effort to make RefPA applicable instead of selecting another method for constructing a reference model.

In addition, the control flow and the activities' property values may not always cause the values for the ranking criteria, as factors like the complexities of application forms between public administrations may influence and distort values for criteria like "Lead Time". Moreover, RefPA does not necessarily detect general best-practice patterns, such as that parallel execution is always better than sequential execution. These patterns must be reflected in the groups' values for the ranking criteria so RefPA can identify them. In addition, it may not always be meaningful to integrate process segments from different public administrations. However, RefPA does not create a reference model in a fully automatic way but supports the user during the construction process. In any case, the user should reflect critically on the appropriateness of RefPA's results and apply his or her knowledge to formulate meaningful queries and interpret the results by, for example, identifying relationships between activities to determine whether the omission of an activity requires other activities to be in place.

Our research makes suggestions for future work. While the recognition of well-known SQL constructs and clear guidance with separated steps and many user interaction points support RefPA's applicability (*NFRQ1*), the evaluation results reveal that RefPA does not meet *NFRQ1*, which is to be addressed in future research. Improved implementations of RefPA should consider our guidelines (see Section 5.2.3) to increase usability and ensure that many process managers can apply the tool in practice. Once RefPA is applied in practice, researchers can perform a more extensive empirical study with a more participants than were part of our evaluation to confirm our results. Researchers can extend RefPA by incorporating new language constructs and operators like LIKE, NOT, and wildcards, which would increase the query language's power and allow users to formulate queries that are more sophisticated than current queries.

Methods for the inductive construction of a reference model are especially relevant to public administrations. This paper proposes RefPA as the first semi-automatic inductive method by which to detect and represent best practices in process models. The results from the evaluation reveal that RefPA is a suitable tool for the construction of reference models in public administrations.

Acknowledgment

This article is a revision of the paper "Semi-Automatic Inductive Derivation of Reference Process Models that Represent Best Practices in Public Administrations" that was presented at the European Conference on Information Systems 2016.

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Symbols

a: An activity
A_i: A finite, non-empty set of activities of model m_i
asc: An atomic *Contains* statement
ash: An atomic *Having* statement
b: A gateway
B_i: A finite set of gateways of model m_i
c_l: The function of a ranking criterion
c_l^*: The adapted function of a ranking criterion
$crit_u$: A tuple of ranking criteria selected by the user with their functions
$crit_u^*$: A tuple of ranking criteria selected by the user with their adapted functions
csc: A combined *Contains* statement that can consist of atomic and other combined statements
csh: A combined *Having* statement that can consist of atomic and other combined statements
c_u: The function of a ranking criterion selected by the user
f: A flow relationship (i.e., an edge)
F_i: The set of flow relationships of model m_i (i.e., edges)
G: The set of all groups that are to be evaluated
gr: A group of activities
k: The length of a sequence of gateways
m_i: A source model
m_m: The merged model
m_r: The reference model
m_e: A function that indicates whether an edge of the merged model has been marked to be transferred to the reference model (1) or not (0)
m_n: A function that indicates whether a node of the merged model has been marked to be transferred to the reference model (1) or not (0)
n: A node (i.e., an activity or gateway)
N_i: The set of nodes of model m_i (i.e., activities and gateways)
oac_u: An operator of an atomic *Containing* statement specified by the user
oah_u: An operator of an atomic *Having* statement specified by the user
occ_u: An operator of a combined *Containing* statement specified by the user
o_l:: A variable that indicates whether the groups are to be ordered ascendingly (1) or descendingly (0) according to the ranking criterion
p: A property
P_i: A finite, non-empty set of properties of model m_i
$props_i$: A mapping that assigns a property value to a pair of one activity and one property in model m_i
p_u: A property selected by the user
S: The set of all source models
sc_u: A *Containing* statement specified by the user
sh_u: A *Having* statement specified by the user
ssc_u: A set of *Containing* sub-statements specified by the user
ssh_u: A set of *Having* sub-statements specified by the user
sub: A sub-statement of a combined statement
s_e: A function that refers to the sources of an edge of the merged model
s_n: A function that refers to the sources of a node of the merged model
v: A value
Val: A property value
V_i: A finite, non-empty set of all property values that occur in model m_i
v_u: A value specified by the user
$\lesssim$: A total preorder

Appendix B. Calculations of ranking criteria

See Table B.1.

Table B.1
Calculations of ranking criteria.

#	Criterion	Calculation
1	Documents	$\lvert\bigcup_{a\in A_{gr}} props_i\,(a, DataObject)\rvert$

2	Change of documents	Algorithmic procedure:
		$Open := \{(a_1, a_2) \in F_i \mid a_1, a_2 \in A_{gr}\}$
		$score := 0$
		For each $f \in Open$:
		If ($\lvert props_i\,(a_1, DataObject)\rvert \geq 1$) Then
		If ($\lvert props_i\,(a_2, DataObject)\rvert \geq 1$) Then
		If ($\lvert props_i\,(a_1, DataObject) \cup props_i\,(a_2, DataObject)\rvert = 0$) Then
		$score := score + 1$
		Else: For each $a_3 \in A_{gr}$ with $(a_2, a_3) \in F_i$:
		$Open := Open \cup \{(a_1, a_3)\}$
		Return *score*

3	Organizational units	$\lvert\bigcup_{a\in A_{gr}} props_i\,(a, Role)\rvert$

4	Change of organizational units	Algorithmic procedure:
		$Open := \{(a_1, a_2) \in F_i \mid a_1, a_2 \in A_{gr}\}$
		$score := 0$
		For each $f \in Open$:
		If ($\lvert props_i\,(a_1, Role)\rvert \geq 1$) Then
		If ($\lvert props_i\,(a_2, Role)\rvert \geq 1$) Then
		If ($\lvert props_i\,(a_1, Role) \cup props_i\,(a_2, Role)\rvert = 0$) Then
		$score := score + 1$
		Else: For each $a_3 \in A_{gr}$ with $(a_2, a_3) \in F_i$:
		$Open := Open \cup \{(a_1, a_3)\}$
		Return *score*

5	Software systems	$\lvert\bigcup_{a\in A_{gr}} props_i\,(a, ITSystem)\rvert$

6	Changes of software systems	Algorithmic procedure:
		$Open := \{(a_1, a_2) \in F_i \mid a_1, a_2 \in A_{gr}\}$
		$score := 0$
		For each $f \in Open$:
		If ($\lvert props_i\,(a_1, ITSystem)\rvert \geq 1$) Then
		If ($\lvert props_i\,(a_2, ITSystem)\rvert \geq 1$) Then
		If ($\lvert props_i\,(a_1, ITSystem) \cup props_i\,(a_2, ITSystem)\rvert = 0$) Then
		$score := score + 1$
		Else: For each $a_3 \in A_{gr}$ with $(a_2, a_3) \in F_i$:
		$Open := Open \cup \{(a_1, a_3)\}$
		Return *score*

7	IT support	$\lvert\{a \in A_{gr} \mid\mid \lvert props_i\,(a, ITSystem)\rvert \geq 1\}\rvert$
8	External contacts	$\lvert\{a \in A_{gr} \mid\mid \lvert props_i\,(a, ExternalStakeholder)\rvert \geq 1\}\rvert$
9	External participants	$\lvert\bigcup_{a\in A_{gr}} props_i\,(a, ExternalStakeholder)\rvert$
10	Legal foundations	$\lvert\{a \in A_{gr} \mid\mid \lvert props_i\,(a, LegalRegulation)\rvert \geq 1\}\rvert$
11	Legally unnecessary steps	$\lvert\{a \in A_{gr} \mid\mid \lvert props_i\,(a, LegalRegulation)\rvert = 0\}\rvert$
12	Media changes	Algorithmic procedure: $Open := \{(a_1, a_2) \in F_i \mid a_1, a_2 \in A_{gr}\}$ $score := 0$ For each $f \in Open$: If ((($\lvert props_i\,(a_1, ITSystem)\rvert \geq 1$) AND ($\lvert props_i\,(a_2, ITSystem)\rvert = 0$)) OR (($\lvert props_i\,(a_1, ITSystem)\rvert = 0$) AND ($\lvert props_i\,(a_2, ITSystem)\rvert \geq 1$))) Then $score := score + 1$ Return *score*
13	Electronic processing	Algorithmic procedure: $score := 0$ For each $a \in A_{gr}$: If (($\lvert props_i\,(a, DataObject)\rvert \geq 1$) AND ($\lvert props_i\,(a, ITSystem)\rvert \geq 1$)) Then $score := score + 1$ Return *score*
14	Paper-based processing	Algorithmic procedure: $score := 0$ For each $a \in A_{gr}$: If (($\lvert props_i\,(a, DataObject)\rvert \geq 1$) AND ($\lvert props_i\,(a, ITSystem)\rvert = 0$)) Then $score := score + 1$ Return *score*
15	Cost	$\sum_{a\in A_{gr}} v \in props_i(a, Cost)$
16	Lead time	$\sum_{a\in A_{gr}} v \in props_i(a, Time)$
17	Processing steps	$\lvert A_{gr}\rvert$
18	Absolute frequency	$\frac{1}{\lvert A_{gr}\rvert} \cdot \sum_{a\in A_{gr}} \lvert\bigcup_{a_m \in A_m \mid a \in Set(s\_n(a_m))} Set\,(s\_n\,(a_m))\rvert$
19	Connectivity	Checks whether the group is connected (return 1) or not (return 0) for instance by using depth-first search

Appendix C. Questionnaire

1. How long have you been involved in projects in public administration?

around _____ years

2. How long have you been involved in conceptual modeling projects?

around _____ years

3. RefPA, with its criteria and query constructs, provided new and valuable suggestions for the resulting reference model.

do not agree		partly agree		fully agree
☐	☐	☐	☐	☐
1	2	3	4	5

4. RefPA's predefined steps gave a reasonable structure to the construction process.

do not agree		partly agree		fully agree
☐	☐	☐	☐	☐
1	2	3	4	5

5. RefPA is useful in constructing the reference model inductively.

do not agree		partly agree		fully agree
☐	☐	☐	☐	☐
1	2	3	4	5

6. RefPA is useful in public administrations' process management.

do not agree		partly agree		fully agree
☐	☐	☐	☐	☐
1	2	3	4	5

7. Users can apply RefPA easily.

do not agree		partly agree		fully agree
☐	☐	☐	☐	☐
1	2	3	4	5

8. As a reference modeler, I would use RefPA again.

do not agree		partly agree		fully agree
☐	☐	☐	☐	☐
1	2	3	4	5

9. As a process manager in a public administration, I would use RefPA again.

do not agree		partly agree		fully agree
☐	☐	☐	☐	☐
1	2	3	4	5

10. What suggestions for improving RefPA do you have?

.....
.....
.....
.....
.....
.....

11. What are the prerequisites for using RefPA in practice?

.....
.....
.....
.....
.....
.....

12. Other comments:

.....
.....
.....
.....
.....
.....

References

[1] C. Houy, P. Fettke, P. Loos, Empirical research in business process management – analysis of an emerging field of research, Bus. Process. Manag. J. 16 (2010) 619–661.

[2] B. Niehaves, R. Plattfaut, J. Becker, Business process management capabilities in local governments: A multi-method study, Gov. Inf. Q. 30 (2013) 217–225.

[3] M. Rosemann, Application reference models and building blocks for management and control, in: P. Bernus, L. Nemes, G. Schmidt (Eds.), Handb. Enterp. Archit., Springer-Verlag, Berlin, Heidelberg, 2003, pp. 595–615.

[4] J. Becker, R. Knackstedt, Konstruktion und anwendung fachkonzeptioneller referenzmodelle im data warehousing, in: W. Uhr, W. Esswein, E. Schoop (Eds.), Wirtschaftsinformatik 2003/Band II Medien - Märkte - Mobilität, Physica-Verlag, Heidelberg, 2003, pp. 415–434.

[5] P. Fettke, P. Loos, J. Zwicker, Business process reference models: Survey and classification, in: C.J. Bussler, A. Haller (Eds.), Proc. BPM 2005 Int. Work. BPI, BPD, ENEI, BPRM, WSCOBPM, BPS, Nancy, 2005, pp. 469–483.

[6] J. Becker, P. Delfmann, R. Knackstedt, Adaptive reference modeling: Integrating configurative and generic adaptation techniques for information models, in: J. Becker, P. Delfmann (Eds.), Ref. Model., Physica-Verlag HD, Heidelberg, 2007, pp. 27–58.

[7] P. Fettke, P. Loos, Classification of reference models: a methodology and its application, Inf. Syst. E-Bus. Manag. 1 (2003) 35–53.

[8] M. Rosemann, W.M.P. van der Aalst, A configurable reference modelling language, Inf. Syst. 32 (2007) 1–23.

[9] O. Thomas, A.-W. Scheer, Tool support for the collaborative design of reference models – A business engineering perspective, in: Proc. 39th Hawaii Int. Conf. Syst. Sci., Kauai, 2006.

[10] J. vom Brocke, Referenzmodellierung: Gestaltung Und Verteilung Von Konstruktionsprozessen, Logos Verlag, Berlin, 2003.

[11] L. Algermissen, P. Delfmann, B. Niehaves, Experiences in process-oriented reorganisation through reference modelling in public administrations - The case study Regio@KomM, in: Proc. 13th Eur. Conf. Inf. Syst., Regensburg, 2005.

[12] M. Karow, D. Pfeiffer, M. Räckers, Empirical-based construction of reference models in public administrations, in: M. Bichler, T. Hess, H. Krcmar, U. Lechner, F. Matthes, A. Picot, B. Speitkamp, P. Wolf (Eds.), Proc. Multikonferenz Wirtschaftsinformatik 2008, Munich, 2008, pp. 1613–1624.

[13] L. Baacke, P. Rohner, R. Winter, Aggregation of reference process building blocks to improve modeling in public administrations, in: A. Grönlund, H.J. Scholl, M.A. Wimmer (Eds.), 6th Int. EGOV Conf. Proc. Ongoing Res. Proj. Contrib. Work., Regensburg, 2007, pp. 149–156.

[14] J. Becker, R. Schütte, A reference model for retail enterprises, in: P. Fettke, P. Loos (Eds.), Ref. Model. Bus. Syst. Anal., Idea Group Publishing, Hershey, 2007, pp. 182–205.

[15] J. Walter, P. Fettke, P. Loos, How to identify and design successful business process models: An inductive method, in: J. Becker, M. Matzner (Eds.), Proc. Rep. PropelleR 2012 Work., Moscow, 2012, pp. 89–96.

[16] J.-R. Rehse, P. Fettke, P. Loos, A graph-theoretic method for the inductive development of reference process models, Softw. Syst. Model. 16 (2017) 833–873.

[17] A. Martens, P. Fettke, P. Loos, Inductive development of reference process models based on factor analysis, in: O. Thomas, F. Teuteberg (Eds.), Proc. 12th Int. Conf. Wirtschaftsinformatik, Osnabrück, 2015, pp. 438–452.

[18] P. Ardalani, C. Houy, P. Fettke, P. Loos, Towards a Minimal cost of change approach for inductive reference process model development, in: Proc. 21st Eur. Conf. Inf. Syst., Utrecht, 2013.

[19] A. Sonntag, P. Fettke, P. Loos, Inductive reference modelling based on simulated social collaboration, in: J.M. Leimeister, W. Brenner (Eds.), Proc. 13th Int. Conf. Wirtschaftsinformatik, WI 2017, St.Gallen, 2017, pp. 701–715.

[20] P.C. Nutt, R.W. Backoff, Organizational publicness and its implications for strategic management, J. Publ. Adm. Res. Theory 3 (1993) 209–231.

[21] L. Algermissen, M. Instinsky, J. Schwall, BPM as a strategic tool for administrative modernization: The IMPROVE approach, in: J. Becker, M. Matzner (Eds.), Proc. Rep. PropelleR 2012 Work., Moscow, 2012, pp. 51–56.

[22] R.-H. Eid-Sabbagh, M. Kunze, M. Weske, An open process model library, in: F. Daniel, K. Barkaoui, S. Dustdar (Eds.), Proc. BPM 2011 Int. Work., Clermont-Ferrand, 2011, pp. 26–38.

[23] A.R. Hevner, S.T. March, J. Park, S. Ram, Design science in information systems research, MIS Q. 28 (2004) 75–105.

[24] K. Peffers, T. Tuunanen, M.A. Rothenberger, S. Chatterjee, A design science research methodology for information systems research, J. Manage. Inf. Syst. 24 (2007) 45–77.

[25] R. Schütte, Grundsätze ordnungsmäßiger Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modelle, Gabler Verlag, Wiesbaden, 1998.

[26] M. Rosemann, A. Schwegmann, P. Delfmann, Preparation of process modeling, in: J. Becker, M. Kugeler, M. Rosemann (Eds.), Process Manag. A Guid. Des. Bus. Process., Springer-Verlag, Berlin, Heidelberg, 2011, pp. 41–89.

[27] C.E. Bogan, M.J. English, Benchmarking for Best Practices: Winning Through Innovative Adaptation, McGraw-Hill, New York, 1994.
[28] H.G. Rainey, R.W. Backoff, C.H. Levine, Comparing public and private organizations, Publ. Adm. Rev. 36 (1976) 233–244.
[29] W.M.P. van der Aalst, Business process management: A comprehensive survey, ISRN Softw. Eng. 2013 (2013) 1–37.
[30] F.D. Davis, Perceived usefulness, perceived ease of use, and user acceptance of information technology, MIS Q. 13 (1989) 319–340.
[31] L. Chung, B.A. Nixon, E. Yu, J. Mylopoulos, Non-Functional Requirements in Software Engineering, Kluwer, Boston, 2000.
[32] J.-R. Rehse, P. Fettke, Towards situational reference model mining – Main idea, procedure model & case study, in: J.M. Leimeister, W. Brenner (Eds.), Proc. 13. Int. Tagung Wirtschaftsinformatik, WI 2017, St. Gallen, 2017, pp. 271–285.
[33] A. Martens, P. Fettke, P. Loos, A genetic algorithm for the inductive derivation of reference models using minimal graph-edit distance applied to real-world business process data, in: D. Kundisch, L. Suhl, L. Beckmann (Eds.), Proc. Multikonferenz Wirtschaftsinformatik 2014, MKWI 2014, Paderborn, 2014, pp. 1613–1626.
[34] M. La Rosa, M. Dumas, R. Uba, R. Dijkman, Business process model merging: An approach to business process consolidation, ACM Trans. Softw. Eng. Methodol. 22 (2013) 11:1–11:42.
[35] J. Leng, P. Jiang, Granular computing–based development of service process reference models in social manufacturing contexts, Concurr. Eng. Res. Appl. 25 (2017) 95–107.
[36] C. Li, M. Reichert, A. Wombacher, Mining business process variants: Challenges, scenarios, algorithms, Data Knowl. Eng. 70 (2011) 409–434.
[37] J.-R. Rehse, P. Fettke, Mining reference process models from large instance data, in: M. Dumas, M. Fantinato (Eds.), Proc. BPM 2016 Int. Work., Rio de Janeiro, 2017, pp. 11–22.
[38] J.-R. Rehse, P. Fettke, P. Loos, An execution-semantic approach to inductive reference model development, in: Proc. 24th Eur. Conf. Inf. Syst., ECIS 2016, İstanbul, 2016.
[39] B.N. Yahya, J.-Z. Wu, H. Bae, Generation of business process reference model considering multiple objectives, Ind. Eng. Manag. Syst. 11 (2012) 233–240.
[40] B.N. Yahya, H. Bae, Generating reference business process model using heuristic approach based on activity proximity, in: J. Watada, G. Phillips-Wren, L.C. Jain, R.J. Howlett (Eds.), Proc. 3rd Int. Conf. Intell. Decis. Technol., Piraeus, 2011, pp. 469–478.
[41] B.N. Yahya, H. Bae, J. Bae, D. Kim, Generating valid reference business process model using genetic algorithm, Int. J. Innov. Comput. Inf. Control 8 (2012) 1463–1477.
[42] P. Fettke, Eine Methode zur induktiven Entwicklung von Referenzmodellen, in: D. Kundisch, L. Suhl, L. Beckmann (Eds.), Proc. Multikonferenz Wirtschaftsinformatik 2014, MKWI 2014, Paderborn, 2014, pp. 1034–1047.
[43] J.-R. Rehse, P. Fettke, P. Loos, Eine Untersuchung der Potentiale automatisierter Abstraktionsansätze für Geschäftsprozessmodelle im Hinblick auf die induktive Entwicklung von Referenzprozessmodellen, in: R. Alt, B. Franczyk (Eds.), Proc. 11th Int. Conf. Wirtschaftsinformatik, Leipzig, 2013, pp. 1277–1291.
[44] F. Gottschalk, W.M.P. van der Aalst, M.H. Jansen-Vullers, Mining reference process models and their configurations, in: R. Meersman, Z. Tari, P. Herrero (Eds.), Proc. OTM 2008 Confed. Int. Work. Posters, ADI, AWeSoMe, COMBEK, EI2N, IWSSA, Monet. OnToContent + QSI, ORM, PerSys, RDDS, SEMELS, SWWS, Monterrey, 2008, pp. 263–272.
[45] B.N. Yahya, H. Bae, J. Bae, Process design selection using proximity score measurement, in: S. Rinderle-Ma, S. Sadiq, F. Leyman (Eds.), Proc. BPM 2009 Int. Work., Ulm, 2010, pp. 330–341.
[46] J. Wang, T. Jin, R.K. Wong, L. Wen, Querying business process model repositories: A survey of current approaches and issues, World Wide Web 17 (2014) 427–454.
[47] M. Kunze, M. Weske, Methods for evaluating process model search, in: N. Lohmann, M. Song, P. Wohed (Eds.), Proc. BPM 2013 Int. Work., Beijing, 2013, pp. 379–391.
[48] R. Dijkman, M. Dumas, B. van Dongen, R. Käärik, J. Mendling, Similarity of business process models: Metrics and evaluation, Inf. Syst. 36 (2011) 498–516.
[49] P. Delfmann, M. Steinhorst, H.-A. Dietrich, J. Becker, The generic model query language GMQL – Conceptual specification, implementation, and runtime evaluation, Inf. Syst. 47 (2015) 129–177.
[50] A. Polyvyanyy, C. Ouyang, A. Barros, W.M.P. van der Aalst, Process querying: Enabling business intelligence through query-based process analytics, Decis. Support Syst. 100 (2017) 41–56.
[51] Y. Ke, J. Cheng, J.X. Yu, Querying large graph databases, in: H. Kitagawa, Y. Ishikawa, Q. Li, C. Watanabe (Eds.), Proc. 15th Int. Conf. Database Syst. Adv. Appl., DASFAA 2010, Tsukuba, 2010, pp. 487–488.
[52] M. Momotko, K. Subieta, Process query language: A way to make workflow processes more flexible, in: A. Benczúr, J. Demetrovics, G. Gottlob (Eds.), Proc. 8th East Eur. Conf. Adv. Databases Inf. Syst., ADBIS 2004, Budapest, 2004, pp. 306–321.
[53] C. Di Francescomarino, P. Tonella, Crosscutting concern documentation by visual query of business processes, in: D. Ardagna, M. Mecella, J. Yang (Eds.), Proc. BPM 2008 Int. Work., Milano, 2009, pp. 18–31.
[54] H. Störrle, VMQL: A visual language for ad-hoc model querying, J. Vis. Lang. Comput. 22 (2011) 3–29.
[55] A. Awad, BPMN-Q: A language to query business processes, in: M. Reichert, S. Strecker, K. Turowski (Eds.), Proc. 2nd Int. Work. Enterp. Model. Inf. Syst. Archit., St. Goar, 2007, pp. 115–128.
[56] C. Beeri, A. Eyal, S. Kamenkovich, T. Milo, Querying business processes with BP-QL, Inf. Syst. 33 (2008) 477–507.
[57] S. Smirnov, H.A. Reijers, M. Weske, A semantic approach for business process model abstraction, in: H. Mouratidis, C. Rolland (Eds.), Proc. 23rd Int. Conf. Adv. Inf. Syst. Eng., London, 2011, pp. 497–511.
[58] J. Becker, L. Algermissen, T. Falk, Modernizing Processes in Public Administrations: Process Management in the Age of E-Government and New Public Management, Springer-Verlag, Berlin, Heidelberg, 2012.
[59] Object Management Group, Business Process Model and Notation (BPMN): Version 2.0, 2011, http://www.omg.org/spec/BPMN/2.0/PDF, (Accessed 23 November 2014).
[60] A.-W. Scheer, O. Thomas, O. Adam, Process modeling using event-driven process chains, in: M. Dumas, W.M.P. van der Aalst, A.H.M. ter Hofstede (Eds.), Process. Inf. Syst. Bridg. People Softw. Through Process Technol., Wiley, Hoboken, 2005, pp. 119–145.
[61] ISO/IEC, ISO/IEC 14977: 1996(e), 1996, http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf, (Accessed 19 January 2018).
[62] E. Rolón, F. Ruiz, F. García, M. Piattini, Applying software metrics to evaluate business process models, CLEI Electron. J. 9 (2006).
[63] A.-M. Sourouni, F. Lampathaki, S. Mouzakitis, Y. Charalabidis, D. Askounis, Paving the way to egovernment transformation: interoperability registry infrastructure development, in: M.A. Wimmer, H.J. Scholl, E. Ferro (Eds.), Proc. 7th Int. Conf. Electron. Gov., Turin, 2008, pp. 340–351.
[64] M.E. Nissen, Redesigning reengineering through measurement-driven inference, MIS Q. 22 (1998) 509–534.
[65] L. Aversano, T. Bodhuin, G. Canfora, M. Tortorella, A framework for measuring business processes based on GQM, in: Proc. 37th Hawaii Int. Conf. Syst. Sci., Waikoloa Village, 2004.
[66] European Commission, The European eGovernment Action Plan 2011-2015: Harnessing ICT to promote smart, sustainable & innovative Government, COM(2010), vol. 743, Brussels, 2010. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2010:0743:FIN:EN:PDF. (Accessed 17 March 2014).
[67] S. Limam Mansar, H.A. Reijers, Best practices in business process redesign: use and impact, Bus. Process Manag. J. 13 (2007) 193–213.
[68] J. Alford, Defining the client in the public sector: A social-exchange perspective, Public Adm. Rev. 62 (2002) 337–346.
[69] J. Becker, P. Bergener, M. Räckers, B. Weiß, A. Winkelmann, Pattern-based semi-automatic analysis of weaknesses in semantic business process models in the banking Sector, in: Proc. 18th Eur. Conf. Inf. Syst., Pretoria, 2010.
[70] P. Delfmann, S. Höhenberger, Supporting business process improvement through business process weakness pattern collections, in: O. Thomas, F. Teuteberg (Eds.), Proc. 12th Int. Conf. Wirtschaftsinformatik, Osnabrück, 2015, pp. 378–392.
[71] M.H. Jansen-Vullers, P.A.M. Kleingeld, M.W.N.C. Loosschilder, H.A. Reijers, Performance measures to evaluate the impact of best practices, in: Proc. Work. Dr. Consort. 19th Int. Conf. Adv. Inf. Syst. Eng., Trondheim, 2007, pp. 359–368.
[72] A. Afonso, L. Schuknecht, V. Tanzi, Public sector efficiency: evidence for new EU member states and emerging markets, Appl. Econ. 42 (2010) 2147–2164.
[73] L. Gelders, P. Mannaerts, J. Maes, Manufacturing strategy, performance indicators and improvement programmes, Int. J. Prod. Res. 32 (1994) 797–805.
[74] R. Diestel, Graph Theory, fourth ed., Springer-Verlag, Heidelberg, 2010.
[75] M. Merz, M.V. Wüthrich, Mathematik für Wirtschaftswissenschaftler: Die Einführung mit vielen ökonomischen Beispielen, Franz Vahlen, München, 2013.
[76] R. Klischewski, Ontologies for e-document management in public administration, Bus. Process Manag. J. 12 (2006) 34–47.
[77] P. Dunleavy, H. Margetts, S. Bastow, J. Tinkler, New public management is dead - long live digital-era governance, J. Public Adm. Res. Theory. 16 (2005) 467–494.
[78] J. Kwon, A. Wellings, S. King, Assessment of the java programming language for use in high integrity systems, ACM SIGPLAN Not. 38 (2003) 34–46.
[79] S. Cass, The 2017 top programming languages, 2017, http://spectrum.ieee.org/computing/software/the-2017-top-programming-languages, (Accessed 4 August 2017).
[80] TIOBE, TIOBE Index for 2017, 2017, https://www.tiobe.com/tiobe-index/, (Accessed 4 August 2017).

[81] H. Scholta, Similarity of activities in process models: Towards a metric for domain-specific business process modeling languages, in: Proc. 24th Eur. Conf. Inf. Syst., İstanbul, 2016.
[82] R. Schütte, J. Becker, Subjektivitätsmanagement bei Informationsmodellen, in: K. Pohl, A. Schürr, G. Vossen (Eds.), Proc. Model., Münster, 1998, pp. 81–86.
[83] A. Koschmider, A. Oberweis, Ontology based business process description, in: M. Missikoff, A. De Nicola (Eds.), Proc. Open Interop Work. Enterp. Model. Ontol. Interoperability, Co-Located with CAiSE'05 Conf., Porto, 2005.
[84] H. Leopold, R.-H. Eid-Sabbagh, J. Mendling, L.G. Azevedo, F.A. Baião, Detection of naming convention violations in process models for different languages, Decis. Support Syst. 56 (2013) 310–325.
[85] F. Pittke, H. Leopold, J. Mendling, Automatic detection and resolution of lexical ambiguity in process models, IEEE Trans. Softw. Eng. 41 (2015) 526–544.
[86] P. Delfmann, S. Herwig, Ł. Lis, Unified enterprise knowledge representation with conceptual models - Capturing corporate language in naming conventions, in: Proc. 30th Int. Conf. Inf. Syst., Phoenix, 2009.
[87] F. Pittke, H. Leopold, J. Mendling, Spotting terminology deficiencies in process model repositories, in: S. Nurcan, H.A. Proper, P. Soffer, J. Krogstie, R. Schmidt, T. Halpin, I. Bider (Eds.), Proc. Int. Work. Bus. Process Model. Dev. Support Int. Conf. Explor. Model. Methods Syst. Anal. Des., Valencia, 2013, pp. 292–307.
[88] F. Gottschalk, W.M.P. van der Aalst, M.H. Jansen-Vullers, Merging event-driven process chains, in: R. Meersman, Z. Tari (Eds.), Proc. OTM 2008 Confed. Int. Conf. CoopIS, DOA, GADA, IS, ODBASE, Monterrey, 2008, pp. 418–426.
[89] J.M. Küster, C. Gerth, A. Förster, G. Engels, A tool for process merging in business-driven development, in: Z. Bellahsene, C. Woo, E. Hunt, X. Franch, R. Coletta (Eds.), Proc. Forum CAiSE'08 Conf., Montpellier, 2008, pp. 89–92.
[90] H.A. Reijers, R.S. Mans, R.A. van der Toorn, Improved model management with aggregated business process models, Data Knowl. Eng. 68 (2009) 221–243.
[91] D.M.M. Schunselaar, E. Verbeek, W.M.P. van der Aalst, H.A. Reijers, Creating sound and reversible configurable process models using CoSeNets, in: W. Abramowicz, D. Kriksciuniene, V. Sakalauskas (Eds.), Proc. 15th Int. Conf. Bus. Inf. Syst., Vilnius, 2012, pp. 24–35.
[92] S. Sun, A. Kumar, J. Yen, Merging workflows: A new perspective on connecting business processes, Decis. Support Syst. 42 (2006) 844–858.
[93] M. Weidlich, R. Dijkman, J. Mendling, The ICoP framework: Identification of correspondences between process models, in: B. Pernici (Ed.), Proc. 22nd Int. Conf. Adv. Inf. Syst. Eng., Hammamet, 2010, pp. 483–498.
[94] M. Weidlich, E. Sheetrit, M.C. Branco, A. Gal, Matching business process models using positional passage-based language models, in: W. Ng, V.C. Storey, J.C. Trujillo (Eds.), Proc. 32th Int. Conf. Concept. Model., Hong-Kong, 2013, pp. 130–137.
[95] H. Leopold, M. Niepert, M. Weidlich, J. Mendling, R. Dijkman, H. Stuckenschmidt, Probabilistic optimization of semantic process model matching, in: A. Barros, A. Gal, E. Kindler (Eds.), Proc. 10th Int. Conf. Bus. Process Manag., Tallinn, 2012, pp. 319–334.
[96] R. Dijkman, M. Dumas, L. García-Bañuelos, R. Käärik, Aligning business process models, in: Proc. 13th IEEE Int. Enterp. Distrib. Object Comput. Conf., Auckland, 2009, pp. 45–53.
[97] C. Klinkmüller, I. Weber, J. Mendling, H. Leopold, A. Ludwig, Increasing recall of process model matching by improved activity label matching, in: F. Daniel, J. Wang, B. Weber (Eds.), Proc. 11th Int. Conf. Bus. Process Manag., Beijing, 2013, pp. 211–218.
[98] M.C. Branco, J. Troya, K. Czarnecki, J. Küster, H. Völzer, Matching business process workflows across abstraction levels, in: R.B. France, J. Kazmeier, R. Breu, C. Atkinson (Eds.), Proc. 15th Int. Conf. Model Driven Eng. Lang. Syst., Innsbruck, 2012, pp. 626–641.
[99] C. Klinkmüller, I. Weber, Analyzing control flow information to improve the effectiveness of process model matching techniques, Decis. Support Syst. 100 (2017) 6–14.