Searching Business Process Repositories Using Operational Similarity

Maya Lincoln and Avigdor Gal

Technion - Israel Institute of Technology
mayal@technion.ac.il, avigal@ie.technion.ac.il

Abstract. Effective retrieval of relevant know-how segments from business process repositories can save precious employee time and support non-expert users in locating and reusing process data. We present a methodology for searching repositories and retrieving relevant process segments, using business logic that is extracted from real-life process models. The analysis of a process repository enables the construction of three taxonomies with which it is possible to process the search intention in operational terms. We tested the method on the Oracle ERP Business Process Model (OBM), showing the approach to be effective in enabling the search of business process repositories.

Keywords: Business process search, Business process repositories, Dynamic segmentation of process models.

1 Introduction

Researchers have become increasingly interested in developing methods and tools for automatically retrieving information from business process repositories [3]. Such repositories are considered important and valuable data reservoirs of organizational know-how. In particular, they enable the retrieval of relevant knowledge segments saving precious time and supporting non-expert users in locating and reusing required process data [24].

Two common methods for retrieving information from a repository are querying and searching. The former is aimed at retrieving structured information using a structured query language. The significance of querying business processes has been acknowledged by BPMI[1] that launched a Business Process Query Language (BPQL) initiative. The latter allows querying information using keywords or natural language and was shown in other areas (e.g. information retrieval) to be an effective method for non-experts.

Research in the field of business process retrieval has mainly focused on *words semantics* and *structural* similarity analysis techniques [3,17,4]. Using these frameworks one can retrieve process models that either contain semantically related components (e.g. activity names with a specified keyword) or match a requested graph structure (e.g. graph paths in which activity A is followed by

[1] Business Process Management Initiative, http://www.bpmi.org/

R. Meersman, T. Dillon, and P. Herrero (Eds.): OTM 2011, Part I, LNCS 7044, pp. 2–19, 2011.

activity B). In this work we tackle the information retrieval challenge from a different angle, with a search framework for business processes that uses operational similarity and enables the retrieval of process *segments* that comply with organizational standards.

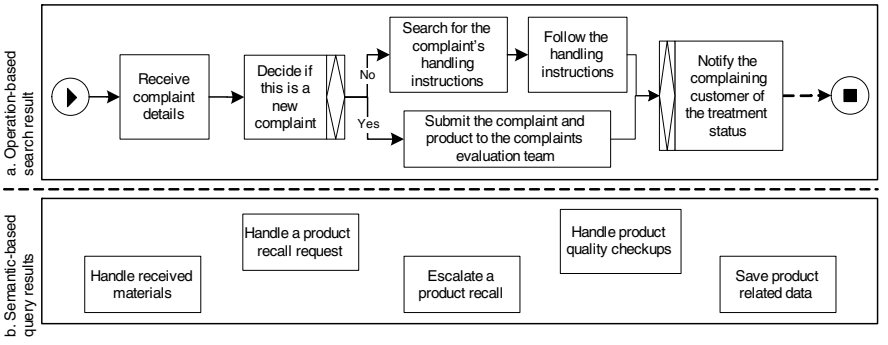

Fig. 1. An example of search results for "how to handle a product recall"

As a motivating example consider an employee interested in finding out "how to handle a product recall." An expected outcome of this retrieval request would be a *segment* from the process repository that represents the order of activities that one should follow in order to achieve the required process goal, as illustrated in Fig 1a. The benefit of such a retrieval framework is that the result is ready for execution. Without any preliminary knowledge of the underlying repository structure, the user can receive a full-fledged process model.

The retrieval output is related to the search phrase in *operational* terms. For example, Fig 1a provides a segment that is only marginally similar to the search phrase text. Specifically, two of the search phrase terms ("Handle" and "Recall") are not represented by any of its activities while the term "Product" is represented by only one activity. Such "how-to" questions are hard to fulfill using common query languages due to the complex logic that is embedded within such questions [4] and especially without specific knowledge on process structure and activity naming. Therefore, using querying techniques, even if they are based on semantic similarity, would likely yield disconnected components, as illustrated in Fig 1b. Such outcome does not tell the user "how-to" fulfill the process goal.

The retrieval framework we propose is based on *operational* similarity. The business logic is extracted from process repositories through the analysis of process activities. Each activity is encoded automatically as a *descriptor* [16]. The collection of all descriptors induces three taxonomies, namely an action scope model, an object grouping model, and an action influence model, with which we can search for the appropriate graph segment, which is the activity flow that provides an answer to the search phrase. The proposed method *dynamically* segments a process repository according to the ad-hoc request as expressed in the user's search phrase.

We present an empirical evaluation based on real-world data, showing that by utilizing the descriptor taxonomies and the process model retrieval method it is possible to effectively support the search of process repositories in operational terms. On average, the best result (that was closest to the expected result) retrieved in each experiment had a 93% precision and was ranked almost always as the first option in the result list.

This work proposes an innovative method for searching business process models while making use of the *how-to* knowledge that is encoded in business process repositories. The following contributions are presented: (a) generic support to an operation-based search of business process models; (b) automatic extraction of business logic from business process repositories; (c) capability to generate ad-hoc process model segments; and (d) an empirical analysis that evaluates the method's usefulness.

The rest of the paper is organized as follows: we present related work in Section 2, positioning our work with respect to previous research. In Section 3 we present the notion of dynamic segmentation in process model repositories. In Section 4 we present an activity decomposition model that is used as the foundation for creating action and object taxonomies. We formulate the search problem and describe our method for searching business process repositories in Section 5. Section 6 introduces our empirical analysis. We conclude in Section 7.

2 Related Work

Related works include query and search techniques in BPM. Works such as [19,20,4,2,9] query business process repositories to extract process model (graph) segments. Such methods require prior knowledge of the structure of the process repository and the exact notation that is used to express it while our work offers techniques that work well even without prior knowledge regarding the process repository.

Keyword search on general tree or graph data structures can also be applied to process repositories [12,10,11]. These methods allow users to find information without having to learn a complex query language or getting prior knowledge of the process structure. Some works extend the tree and graph keyword search methods to support a more intuitive interface for the user by enabling searches based on natural language [14,13]. According to [1], the straightforwardness of a natural language makes it the most desirable database query interface. The retrieved information in both keyword and natural language search methods is in the form of single process model components such as activities and roles. Our work extends such works to support the retrieval of complete segments by applying dynamic segmentation of the process repository. The search result is a compendium of data (a segment of a business process model) related to the operational meaning of the searched text.

Another line of work focuses on automatic construction of process data ontologies. The work in [5] proposes a query-by-example approach that relies on ontological description of business processes, activities, and their relationships,

which can be automatically built from the workflow models themselves. The work in [6] automatically extracts the semantics from searched conceptual models, without requiring manual meta-data annotation, while basing its method on a model-independent framework. Our framework differs from these works in that we target the automatic extraction and usage of the operational layer (the "how-to") and the business rules encapsulated in the process repository.

3 Dynamic Segmentation

This section describes dynamic segmentation in process model repositories. Section 3.1 presents a model for business process repositories followed by a description and a formal model of a dynamic segmentation.

3.1 A Business Process Repository and Static Segmentation

The Workflow Management Coalition (WFMC) [7] defines a *business process model* as a "set of one or more linked procedures or activities which collectively realize a business objective or policy goal." An example of such a business process model is the "Fulfill a Purchase Order" process model, presented in Fig. 2a using YAWL [22]. The dashed lines represent parts of the process that are not shown due to space limitations.

A *process repository* is a collection of interconnected process models, where the execution of an activity in one process model may invoke the execution of another activity in another process model. Formally, we define a process repository to be a graph $G = (V, E)$, where V represents activities and an edge (a_i,a_j) in E exists if a_j's execution follows that of a_i.

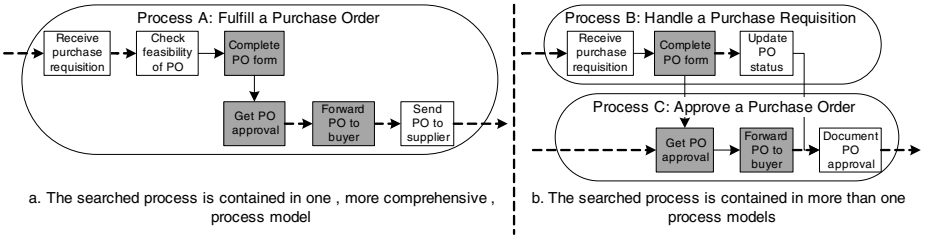

Fig. 2. Examples of process models

Each of the process models in the process repository has a name, and we refer to it as a *static* segment of the process repository. For example, the processes "Handle a Purchase Requisition" in Fig. 2b represents a segment of process activities (steps). This static, pre-defined, segmentation of process repositories is determined according to the logic of the repository developer and changes from one repository to the other. To illustrate, the SAP business process repository consists of more than 16,500 activities segmented into more than 2,400 processes

while the Oracle business process repository (OBM) consists of about 9,700 activities segmented into 1,553 processes. 92% of the two repositories refer to the same business areas (e.g. processes from the domain of accounting, human resource management, production, logistics, etc.), yet only 23% of the process names are semantically similar. The rest of the processes have segments that do not share starting and/or ending activities.

3.2 Dynamic Segments in Business Process Repositories

It is possible to partition the process repository graph in different ways, creating new, *dynamic*, process model segments. To illustrate, consider a process for manually creating a purchase order. In one repository, this process can be part of a larger process model called: "Fulfill a Purchase Order," which includes the creation, approval, and submission of a purchase order (see illustration in Fig. 2a, where the dynamic segment activities are shaded). In a different process repository, with different process models, the sequence of activities that fulfill the example process goal cut across two static process models: "Handle a Purchase Requisition" and "Approve a Purchase Order" (see illustration in Fig. 2b). Note that although in both repositories there is an activity sequence that fulfills the searched process goal (Manually Create a Purchase Order) - this process model is not represented separately (as a standalone segment) in those repositories and therefore does not have a pre-defined process name.

Formally, we can define a dynamic segment to be a sub-graph induced by G. Let $\mathcal{G}$ be the set of all subgraphs that are induced by G, and hence, the set of all possible dynamic segmentations of a repository.

$$\mathcal{G} = \{G' = (V', E') | V' = \{v_1, v_2, ...v_n\} \epsilon 2^{|V|} \wedge \forall \{v_i, v_j\} \epsilon V', (v_i, v_j) \epsilon E' \text{ if } (v_i, v_j) \epsilon E\}$$

4 Descriptor Analysis

This section enhances the descriptor model of [16,15] to support process model search. We provide a brief overview of the descriptor model in Section 4.1 followed by an introduction of three new taxonomies in sections 4.2-4.4. To illustrate and assess the taxonomies we use the Oracle Applications ERP process repository.

4.1 The Descriptor Model

In the Process Descriptor Catalog model ("PDC") [16] each activity is composed of one action, one object that the action acts upon, and possibly one or more action and object qualifiers. Qualifiers provide an additional description to actions and objects. In particular, a qualifier of an object is roughly related to an object state. State-of-the-art Natural Language Processing (NLP) systems, *e.g.*, the *Stanford Parser*,[2] can be used to automatically decompose process and activity names into *process/activity descriptors*. Each descriptor, d, can therefore

[2] http://nlp.stanford.edu:8080/parser/index.jsp

be represented as a tuple $d = (o, oq, a, aq)$, where o is an object, oq is a set of object qualifiers, a is an action, and aq is a set of action qualifiers.

For example, in Fig. 1, the activity "Notify the complaining customer of the treatment status" generates the following activity descriptor: ("customer", "complaining", "notify", "of the treatment status").

We denote by A the set of all actions (including qualifiers) in a process repository, G. Similarly, O denotes the set of all objects. We also denote by $a(d)$ the action part of the descriptor, including the action qualifiers, e.g., "notify of the evaluation status" in the example. Similarly, $o(d)$ denotes the object part.

4.2 The Action Scope Model (ASM)

The action scope model is a graph $ASM = (V_{ASM}, E_{ASM})$, which represents the relationship between an action in a process name (a *primary action*) and the actions in its corresponding static model segment. As such, it represents an operational meaning of primary actions in the repository. Recall that a process repository consists of pre-defined process segments. We use this segmentation for learning about the scope of actions in the following way. Each action in the repository is related with a set of directional graphs of actions that represent the order of actions within this primary action's segments. Therefore, V_{ASM} is a set of descriptor actions that are found in the segments of the primary action. An edge in E_{ASM} connects two actions that appear sequentially in the process model. Since such a primary action can be part of more than one process name, and since the same action may be represented more than once in the same process model segment - each edge in the action scope model is labeled with its weight, denoted $w(e)$, calculated by the number of its repetitions in the related process model segments. Graph splits are also represented in the action scope model. We denote by $create_{ASM}(PM)$ the action of creating an ASM from a given static model segment, PM.

As a motivation for using this model in the context of searching process models, we analyzed 17 real-life processes from the Oracle Business Model (OBM) (we elaborate on the experiment setup in Section 6). Based on this sample we calculated a *tf-idf* measure [18] for each action as follows. For a group of processes that share the same primary action, we count the number of times each action appears in the group (serves to compute *tf*) and the number of different processes in the repository (processes with a different primary action) where this action appears overall (serves to compute *idf*). For example, out of the 17 processes, five contain the primary action "Create," including 49 activities altogether. The action "Define" is presented five times in the set of all actions: four times in the "Create" processes, and one time in a different process. Therefore, its *tf-idf* is $(4/49) * (log(13/(1+2))) = 0.052$. Note that the five "Create" processes are taken into account as one large process for this calculation.

In order to examine the quality of this value, we compare it to a random *tf-idf* values as follows. When assuming a random distribution of activities over processes, an activity that repeats n times in the repository has an expected n/k appearances in each process, assuming k processes. Therefore, given m processes

that share the same primary action, we expect an activity to appear $n*m/k$ times in these processes, and the *tf-idf* computation is done accordingly. Continuing with the previous example, with five appearances of "Define" in the repository and 17 processes, we expect "Define" to appear 5/17 times in a process and $5 * 5/17 = 1.47$ times in "Create" processes and 3.53 times elsewhere. This results in a tf-idf of 0.01. When applying such a calculation to all actions in all the 17 processes, it was found that on average, their *tf-idf* is 4.6 times better than the random value. We are therefore encouraged to learn about participating actions in a process segment given a primary action.

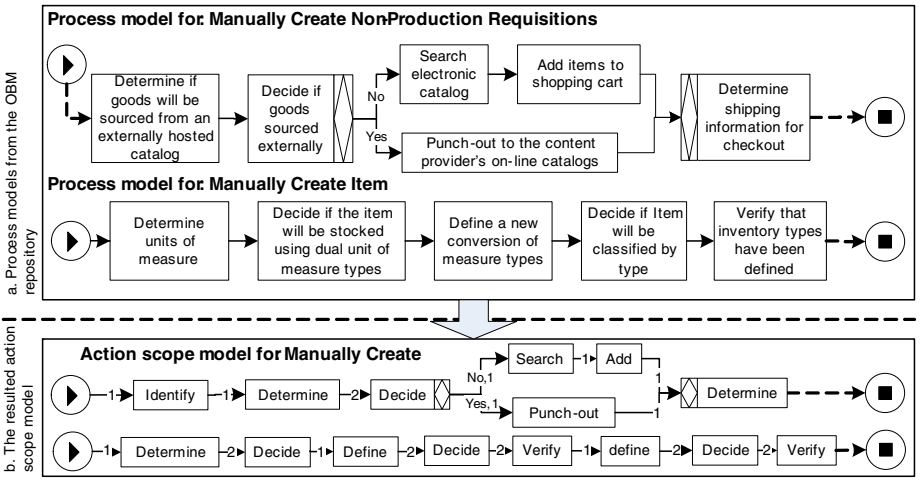

Fig. 3. A segment of the action scope model for the action "Manually Create" in the OBM repository for the Procurement category

Consider the following two processes from the OBM repository: "Manually Create Non-Production Requisitions" and "Manually Create Item." These processes are represented in the OBM by corresponding graph segments as illustrated in Fig 3a. Using these two process models, it is possible to generate an action scope model for the action "Manually Create" (Fig 3b). The dashed lines in this illustration represent graph parts that are not shown due to space limitations. According to this example, there are two optional action paths compatible to the "Manually Create" action starting by either "Identify" or "Determine." Since "Decide" follows "Determine" twice in this model, the respective edge weight is set to 2.

Given an action scope model graph, ASM, the intensity of any path i of length $n-1$, $(a_{i,1}, a_{i,2}, ..., a_{i,n})$ in ASM, ι_i^{ASM}, is computed to be:

$$\iota_i^{ASM} = 1 - \frac{n-1}{\sum_{j=1}^{n-1} w(a_{i,j}, a_{i,j+1})}.$$

The values of ι_i^{ASM} are in [0,1], with higher values representing a stronger, more common ASM_i. For example, the intensity of the second path in the action scope model for "Manually Create" (Fig. 3b) is calculated as follows:

$$\iota_2^{ASM} = 1 - \frac{8}{1+2+1+2+2+1+2+2} = 0.385.$$

We define *fitness*, f^{ASM}, between a segment, PM, and an ASM to be zero if there is no path in the ASM that represents the PM. If such path exists, the *fitness* value is determined by its intensity. Formally, we define *fitness* as follows:

$$f^{ASM}(PM, ASM) = \begin{cases} \iota_i^{ASM} & \exists\ ASM_i \epsilon ASM \mid create_{ASM}(PM) = ASM_i \\ 0 & otherwise \end{cases}$$

4.3 The Object Grouping Model (OGM)

The object grouping model represents the relationship between an object (a *primary object*) and the objects in its corresponding static model segment. As such, it represents an *operational* meaning of primary objects in the repository. Since such a primary object can be part of more than one process segment, and since the same object may be represented more than once in the same process model segment - each edge in the object grouping model is labeled with its weight calculated by the number of its repetitions in the related process model segments. Therefore, an object grouping model of an object, o, denoted as $OGM(o)$ is a bag (repetitions allowed) $\{o_1, o_2, ..., o_k\}$.

As a motivation for this model usage in the context of this work, and similarly to the *tf-idf* calculations made for assessing the relevance of using the ASM, we calculated the *tf-idf* measure for each object in each group of processes that share the same primary object. As a result, we found that on average, their *tf-idf* is 5.3 times better than the random value. We are therefore encouraged to learn about participating objects in a process segment given a primary object.

To illustrate, consider two processes from the OBM repository: "Issue a Purchase Order" and "Authorize a Purchase Order." These processes are represented in the OBM by corresponding graph segments as illustrated in Fig 4a. Using these two process models, it is possible to generate an object grouping model for the object "Purchase Order," as illustrated in Fig 4b. According to this example, process models that are related to the object "Purchase Order" deal also with objects such as "Purchase Order Notifications," "Blanket Agreement Document," and "Requisitions," while the last option's weight equals 2 since it is represented twice in the two input process models.

Given a segment, PM, (possibly a result of a user's search phrase) we calculate its proximity, λ^{OGM}, to a given object grouping model, OGM, as follows. We denote by OPM the set of all objects in PM. We compute precision and recall on OPM and $OGM(o)$ as follows:

$P(PM, OGM(o)) = \frac{|OPM| \cap |OGM(o)|}{|OPM|}$; $R(PM, OGM(o)) = \frac{|OPM| \cap |OGM(o)|}{|OGM(o)|}$. PM is considered more similar to $OGM(o)$ as both $P(PM, OGM(o))$ and

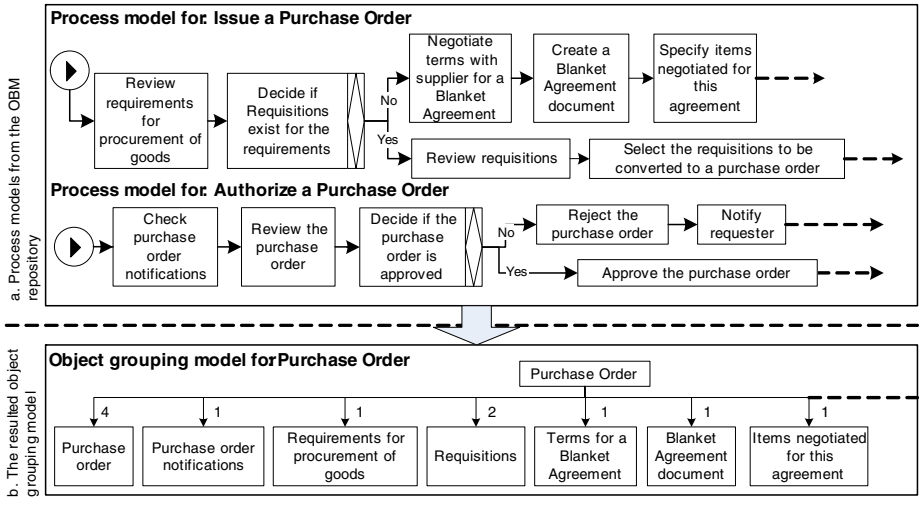

Fig. 4. A segment of the object grouping model for "Purchase Order" in the OBM repository for the Procurement category

$R(PM, OGM(o))$ are higher. Therefore we define $\lambda^{OGM}(PM, OGM(o))$ as the harmonic mean of these figures:

$$\lambda^{OGM}(PM, OGM(o)) = \frac{P(PM, OGM(o)) * R(PM, OGM(o))}{P(PM, OGM(o)) + R(PM, OGM(o))} * 2$$

λ^{OGM} ranges between [0,1], and higher values represent a stronger match between the given PM and $OGM(o)$. It is also possible to define a threshold, th^{OGM}, below which λ^{OGM} is set to zero.

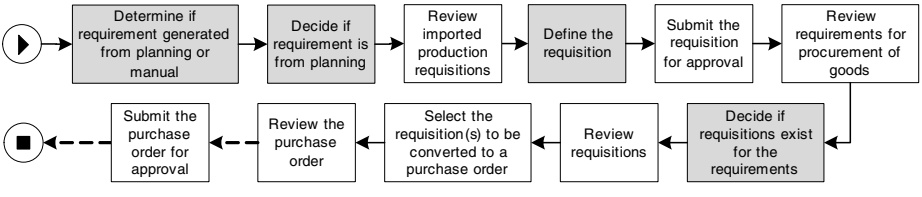

Fig. 5. An example of a process model for "Manually create purchase order"

For example, let us examine a given process model PM that represents the flow of activities for "Manually create purchase order," as illustrated in Fig. 5. The object grouping model for "Purchase Order," (Fig. 4) and PM share three objects in common: "Purchase order," "Requisition," and "Requirements for procurement of goods," that repeat in PM activities once, twice, and once again, respectively (see Fig. 6). Both models consist of 11 objects, each. Therefore,

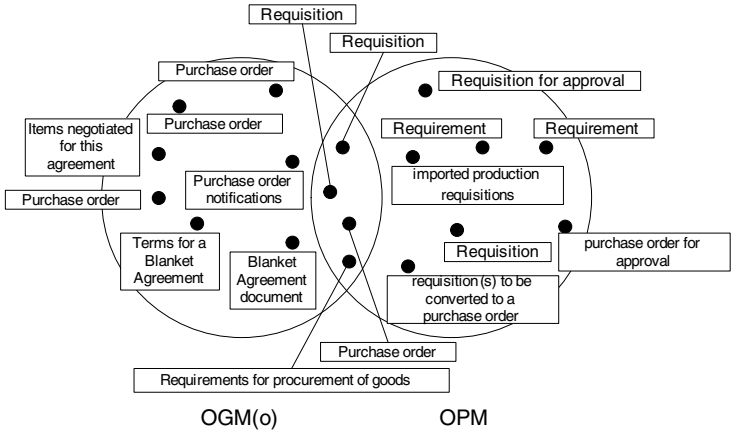

Fig. 6. An example of proximity calculation

precision and recall are both 4/11 (single and plural forms of the same object are considered to be the same) and the proximity in our example is calculated to be 0.18.

4.4 The Action Influence Model (AIM)

The action influence model represents the relationship between a primary action and the flow of states (object qualifiers) of the primary object in static model segments that correspond to the primary action. As such, it reflects the influence of a primary action on the way a primary object changes. Each edge in the action influence model is labeled with its weight representing the number of its repetitions in the related process model segments.

As a motivation for this model usage in the context of this work, and similarly to the calculations made for the *ASM*, we calculated the *tf-idf* measure for each two adjacent object-qualifiers (object-qualifier pairs) in any of the static model segments that share the same primary action. We found that on average, their *tf-idf* is 3.6 times better than the random value. We can therefore deduce that it is possible to learn about participating object-qualifier pairs in a process segment given a primary action.

To illustrate, consider the two process models named: "Manually Create Non-Production Requisitions" and "Manually Create Item." They both deal with *manual creation,* but focus on different objects, "Non-Production Requisitions" and "Item." Their initial part is illustrated in Fig. 3a and continued in Fig. 7a. By following changes to the qualifiers of the primary object in these process models we end up with the action influence model for "Manually Create" as illustrated in Fig. 7b. In this example, both primary object states change from "For approval" to "New" and then to "Rejected" or "Approved" in their corresponding process models and therefore the corresponding edges in the action influence model are labeled with weight of 2. In addition, we note that one of the

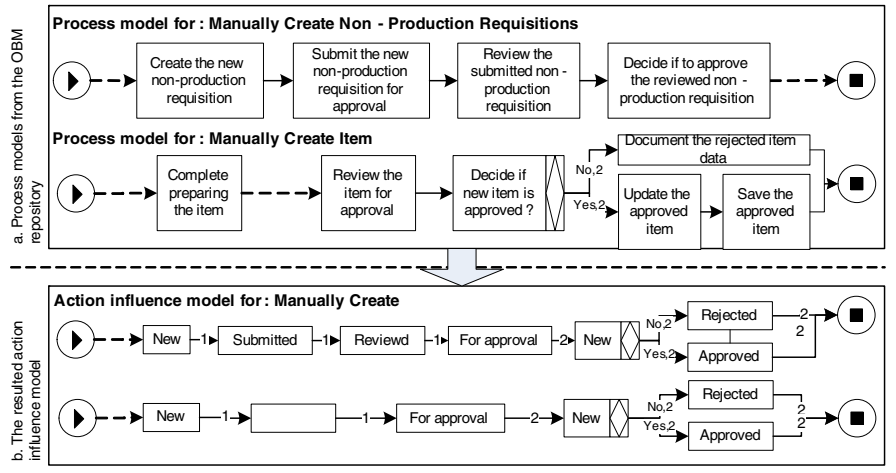

Fig. 7. A segment of the action influence model for the action "Create" in the OBM repository for the Procurement category

qualifiers in the action influence model is represented by an empty rectangle since its corresponding activity included the object "Item" as is, without any qualifiers. Also, it is worth noting that the first two activities in the process model for "Manually Create Non-Production Requisitions" represent the object "New Item." Nevertheless, the corresponding qualifier "New" is represented only once at the beginning of the action influence model, since no change has been made to the object "Item" when advancing the process between these two activities.

Given a process model, PM, it is possible to calculate its similarity, σ_i^{AIM}, to a given path in an action influence model, AIM_i, using one of the state-of-the-art methods for assessing similarity between process models (e.g. [23,8,21]). For that purpose, an action influence model is created for the primary object of PM (referred to as a temporary action influence model, $TAIM$) and then compared to AIM_i. σ_i^{AIM} can be then normalized to the range of [0,1]. On top of this score for each path in AIM, we also add an additional score that reflects the weights of the matched edges, so that the proximity between PM and AIM, λ^{AIM}, is calculated as follows:

$$\lambda^{AIM}(PM, AIM) = \frac{\Sigma_{i=1}^{k}(\sigma_i^{AIM} + \frac{\sum_{e \epsilon AIM_i \cap TAIM} {}^{w(e)}}{\sum_{e \epsilon AIM_i} {}^{w(e)}})}{k} * \frac{1}{2},$$

where $w(e)$ is the weight assigned to e in AIM_i, and k is the number of paths in AIM. Note that since the additional score refers only to edges, it can be zero even when $\sigma_i^{AIM} > 0$ (e.g. when some activity names are matched) and therefore it is not multiplied, but added to σ_i^{AIM}.

λ^{AIM} ranges between [0,1], where higher values reflect a higher proximity of the given PM and AIM. It is also possible to define a threshold, th^{AIM}, in a similar manner as th^{OGM}.

Following our example, we first generate a temporary AIM for "Manually Create" based on the example process model graph (Fig. 5). Since the primary object "Purchase Order" is presented twice in this process model, at first without qualifiers, and then with the qualifier "For approval," the temporary AIM contains these two qualifiers, as illustrated in Fig. 8.

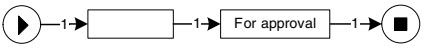

Fig. 8. An example of a temporary AIM

At the next phase of our example, we calculate the similarity between the temporary AIM and each of the two object qualifier paths presented in the action influence model for "Manually Create" (Fig. 7b). To do that we use the similarity method presented in [23]. There is a full match (similar text) of one node ("For approval") between the temporary AIM and the first path in the AIM for "Manually Create," and a full match of two nodes between the temporary AIM and the second path in the AIM for "Manually Create." Therefore, the similarity score that these matches contribute is 1 and 2 for the first and second path, respectively. Since the optimal match between the temporary AIM and each of the paths in the AIM for "Manually Create" is a situation in which the entire temporary AIM is fully matched, the optimal similarity score is 2. Therefore, the similarity score between the temporal AIM and the first and second paths is $\sigma_1 = \frac{1}{2}$ and $\sigma_2 = \frac{2}{2} = 1$, respectively. In addition, there are no matched edges between the temporary AIM and the first path (since the node labeled "For approval" is not directly connected to the start or end nodes). Nevertheless, there is one matched edge between the temporary AIM and the second path - that directly connects the node with no qualifiers with the "For approval" node. This edge's occurrence is 1 and its weight is 1, resulting in a score addition of $\frac{1}{12}$ to the score of the second path. This addition is divided by the maximal number of matched edges (3 in this example). As a result, the final similarity score is: $\lambda^{AIM}(PM, AIM) = \frac{(\frac{1}{2}+0)+(1+\frac{1}{12})}{2} * \frac{1}{2} = 0.4$.

5 The Process Model Search Problem and Method

This section describes the process model search method. We first provide a formal description of the search problem (Section 5.1). Then, we describe in Section 5.2 the use of the three taxonomies of Section 4 in a search procedure.

5.1 The Process Model Search Problem

Let G be a graph that represents a process repository. Let λ be a benefit model $\lambda : \mathcal{G}, A, O \rightarrow [0, 1]$ and let S be a search phrase, given either as a descriptor or in natural language that is converted into a descriptor (see Section 4.1).

Given S and a segment G' we define λ as follows:

$$\lambda(G', a(S), o(S)) = f(G', ASM(a(S)) * \lambda^{OGM}(G', OGM(o(S)) * \lambda^{AIM}(G', AIM(a(S)) \quad (1)$$

Given S, G, and λ find a segment, G', that best fits the search phrase. The process model search problem can be formally defined as follows:

Problem 1. Given G, S, and λ, find $G' \epsilon \mathcal{G}$ s.t. $G' = argmax_{G'' \epsilon G} \lambda(S, G'')$

In what follows we actually find the top k segments, ranked according to their operational relevance to the searched phrase. This way we expand the retrieved result range, allowing users to examine more than one result that may also contain useful information with regards to the search problem.

5.2 The Process Model Search Method

The process model search method (PMSM) relies on an underlying process descriptor analysis model and dynamically segments a business process repository to fit a given search phrase. We use the search request *"Manually Create a Purchase Order"* to illustrate the suggested method.

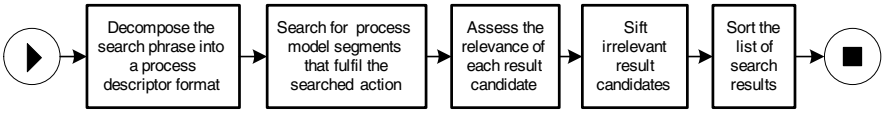

Fig. 9. The process model search method

Method Overview. The search procedure is composed of five main phases as follows (see illustration in Fig. 9). At first, it receives as input the name of a required process model in natural language or as a process descriptor. For the former, the input is automatically decomposed into a process descriptor format using NLP systems such as the *Stanford Parser*. According to this phase, the search phrase in our example will be transformed into the following process descriptor: object="order," action="create," object qualifier="purchase," action qualifier="manually." If more than one action and one object appear in the search phrase, it is automatically interpreted as separate descriptors (expressing all possible combinations of the given descriptor components) that are separately searched. Handling search phrases that include multiple process names as a unified process group is a topic for future work.

Based on the process descriptor input (the "target descriptor"), the PMSM searches first for all process model segments within the process repository that can be relevant candidates for the search result. This dynamic segmentation phase is the most important phase in this method since it enables the discovery of process models that do not have a pre-defined static segment in the repository, but are rather a part of or a sequential collection of pre-segmented process models (see Section 3.2).

At the next phase each process model option is assessed according to three orthogonal measures that reflect its relevance to the search request using proximity to the action scope model, the object grouping model, and the action influence model. According to these three relevance measures, non-relevant process models are removed from the option list, and finally the remaining process model options are being sorted according to a weighted grade of the three relevance measures. The sorted process model segment list is presented to the requester - as the search result for her request.

Phase 1: Dynamic Segmentation. The goal of this phase is to retrieve all process model segments that fulfill the target **action**. For example, given the target action "Manually Create," the PMSM will search at this phase all process model segments that are aimed at *manually creating* an object of any type, without restricting the segment to the target object. Note that a naïve solution in this phase would be to examine all possible process model segments within the repository. Nevertheless, such algorithm can be highly inefficient (with n static segments and m the size of the biggest segment, there are 2^{nm} induced dynamic segments over G). Therefore, we reduce the collection of segments by selecting only relevant candidates at the first phase. Also note that this phase focuses on actions rather than objects as a basis for retrieving optional search results, since the search method is based on the operational meaning of the search and therefore all process model segments that relate to the search action represent a full set of optional results from which we select relevant results that are also related to the searched object.

To do that, the PMSM searches in the process repository for graph segments that are similar - both *structurally* and *textually* - to the action scope model of the target action. Since the action scope model is a graph of activities with partial names (only the action part is represented) - it is possible to convert it into a query statement using any state-of-the-art business process query mechanisms (e.g. [2,4]) and search for graph segments that are similar to it.

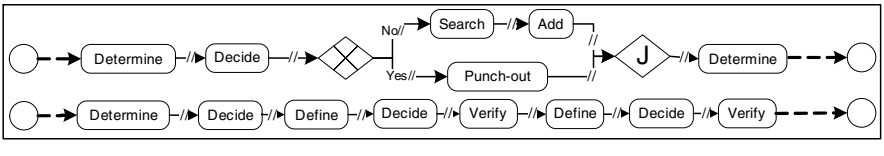

Fig. 10. Segments of *BPMN-Q* queries generated from the ASM for "Manually Create" in the OBM repository

In our example, the action scope model for "Manually Create" consists of two graphs (see Fig. 3). Each of these graphs will be converted into a query statement and be searched separately in the repository. A representation of those two graphs as a query statement using the *BPMN-Q* method suggested in [2] is illustrated in Fig. 10. The dashed lines in this illustration represent query parts that are not presented here due to space limitations. To relax the generated

query statements we mark all their edges with "//" - stating that there may be zero or more activities between each connected activities in the query result.

By running the two queries of our example on the OBM repository using the *BPMN-Q* method, 14 results were retrieved. One of those results is presented in Fig. 5. This example highlights the need for dynamic segmentation. This segment is combined sequentially from three process models in the OBM repository - starting with the last five activities of "Create Production Requisitions," continuing with "Issue a Purchase Order" (Fig. 4a) and terminating with the first activities of "Authorize a Purchase Order" (also presented in Fig. 4a).

Phase 2: Assessing the Relevance of Result Candidates. This phase is aimed at determining the relevance of each result option retrieved at the previous phase to the search request. To do that, we use the three measures defined in Section 4 as follows.

1. Each result retrieved at phase 1 is compatible with one of the action paths in the action scope model, and therefore related to the intensity, i_j of this path (as detailed in Section 3).
2. We calculate each result's λ^{OGM} (Section 4.3) as a measure for its proximity to the target object. To understand the necessity of this measure consider, for example, a process segment resulted at phase 1 that highly represents the "Manually Create" action (the order of its actions represents one of the action sequences in the *ASM* of the "Manually Create" action). Nevertheless, none of the objects involved in this process model participates in process models related to the object "Purchase Order." In this case we can deduce that this segment is only loosely related to the primary object in our example.
3. The proximity to the action influence model is aimed at preferring results that express a *typical* modification of the primary object states as a result of applying the primary action. To measure the proximity between each result to the primary action's influence model we use λ^{AIM} (Section 4.4).

Phase 3: Sifting Irrelevant Result Candidates. In this phase two thresholds, th^{OGM} and th^{AIM} (see sections 4.3 and 4.4), are used to determine the inclusion of each result candidate in the final result list. Note that it is not sufficient to apply a threshold on the final grade (as calculated in phase 4) to determine inclusion. A candidate may receive a relatively high final grade and still be irrelevant to the searched phrase. For example, a candidate may be highly similar to the search phrase's action influence model but may not consist any relevant objects from its object grouping model. Therefore, we include in the final result list only results for which $\lambda^{OGM} > th^{OGM}$ and $\lambda^{AIM} > th^{AIM}$.

Phase 4: Sorting the List of Search Results. At this phase a final grade, λ_i, for each result candidate, R_i, is calculated according to Eq. 1, using the grades of the three measures calculated in phase 2.

Following our example, the final grade for the optional result presented in Fig. 5 is: $\lambda_i = 0.385 * 0.18 * 0.4 = 0.03$. Finally, the optional process models are

sorted according to their final grade in an ascending order - from the closest to the most distant option as regards to the user's search phrase.

6 Experiments

We now present an empirical evaluation of the proposed method effectiveness. We first present our experimental setup and describe the data that was used. Then, we present the experiment results and provide an empirical analysis of these results.

Data Set and Experiment Setup. We chose a set of 17 real-life processes from the Oracle Business Model (OBM),[3] comprising 152 activities altogether. Using these processes we created a "process repository database," that includes the 17 process models and their derived taxonomies.

To evaluate the suggested method we conducted 17 experiments. At each experiment, a single process was removed from the database and then was searched according to its name. More specifically, each experiment was conducted according to the following steps: (a) preparation: removal of one of the process names from the database and reconstruction of the taxonomies so that the database will contain the process model (its graph segment) but the three taxonomies will not contain any of its descriptor components; (b) search for the removed process in the database, using its name as the search phrase.

The similarity between the result's temporal AIM and each of the object qualifier paths in the primary action's AIM (see Section 4.4) was calculated using the method in [23]. In addition, the threshold parameters for sifting irrelevant result candidates (defined in Section 5.2) were set to: $th^{OGM} = 0.1$, $th^{AIM} = 0.2$.

The search results have then been objectively evaluated with respect to the original, removed, process. For each of the 17 experiments we also chose the "best result," the one most similar to the goal process model, calculated using the similarity method presented in [23]. Our metrics for measuring the effectiveness of the method, as detailed below, assess both the quality of an average result, as well as the average quality of the best result in each experiment - showing the best performance of the search method.

We use five metrics to measure the effectiveness of the method. For all results and for the best result we compute: (1) the average percentage of correct activities, showing how similar the results are with respect to the goal, correct result; (2) the average percentage of redundant activities, showing the amount of redundant, unnecessary activities in the retrieved results. Finally, we identify the average location of the best result in the list of search results - showing the effectiveness of the grading mechanism, that aims at locating the best result at the beginning of the result list.

Results and Analysis. On average, each result contained 81.7% of the goal process model activities. On average, the best result has a higher overlap of

[3] http://www.oracle.com/applications/tutor/index.html

92.9% with the goal process model and was ranked in a high location in the list of results (1.2), usually in the first place. In addition to the correct activities, the results also contained, on average, 14.2% of redundant activities out of the goal process model, while this percentage was lower (8.0%) in the best result.

These experiments have shown the usefulness of using a descriptor repository in searching for business process models based on their operational meaning. We also showed the method to be effective in the given experimental setup, both in terms of the similarity between the results and the goal result and with respect to the ranking of the best result.

7 Conclusions

We proposed a mechanism to automate the search of process models that saves search time and supports non-expert users in searching for business process models in a process repository using their own terminology of the process goal. By analyzing a process repository we can automatically create three taxonomies with which it is possible to process the search intention in operational terms. We provide a formal model of the problem, a method for providing a ranked response to a user's request, and we show the method to be empirically effective, using the Oracle ERP Business Process Model, as a testbed.

The proposed method and experiments provide a starting point that can already be applied in real-life scenarios, yet several research issues remain open. We mention four such extensions here. First, extending the empirical study to further examine the quality of retrieved search results aiming at improving and fine-tuning the method. Second, supporting search phrase relaxations to retrieve more result options in cases where the search phrase retrieves only few or no results. Third, extending the method for supporting synonyms to extend the repository vocabulary. Forth, supporting multiple descriptors in a search phrase.

References

1. Androutsopoulos, I., Ritchie, G.D., Thanisch, P.: Natural language interfaces to databases–an introduction. Natural Language Engineering 1(01), 29–81 (1995)
2. Awad, A.: BPMN-Q: A Language to Query Business Processes. In: EMISA 2007, vol. 119, pp. 115–128 (2007)
3. Awad, A., Polyvyanyy, A., Weske, M.: Semantic querying of business process models. In: 12th International IEEE Enterprise Distributed Object Computing Conference, pp. 85–94. IEEE (2008)
4. Beeri, C., Eyal, A., Kamenkovich, S., Milo, T.: Querying business processes with BP-QL. Information Systems 33(6), 477–507 (2008)
5. Belhajjame, K., Brambilla, M.: Ontology-based description and discovery of business processes. Enterprise. Business-Process and Information Systems Modeling, 85–98 (2009)
6. Bozzon, A., Brambilla, M., Fraternali, P.: Searching Repositories of Web Application Models. In: Benatallah, B., Casati, F., Kappel, G., Rossi, G. (eds.) ICWE 2010. LNCS, vol. 6189, pp. 1–15. Springer, Heidelberg (2010)

7. Coalition, W.M.: The workflow management coalition specification - terminology & glossary. Technical report, Technical Report WFMC-TC-1011, Workflow Management Coalition (1999)
8. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring similarity between semantic business process models. In: Proceedings of the fourth Asia-Pacific Conference on Comceptual Modelling, APCCM 2007, pp. 71–80. Australian Computer Society, Inc., Darlinghurst (2007)
9. Goderis, A., Li, P., Goble, C.: Workflow discovery: the problem, a case study from e-Science and a graph-based solution (2006)
10. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: XRANK: Ranked keyword search over XML documents. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 16–27. ACM (2003)
11. He, H., Wang, H., Yang, J., Yu, P.S.: BLINKS: ranked keyword searches on graphs. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, pp. 305–316. ACM (2007)
12. Hristidis, V., Papakonstantinou, Y., Balmin, A.: Keyword proximity search on XML graphs (2003)
13. Katz, B., Lin, J., Quan, D.: Natural language annotations for the Semantic Web. In: Meersman, R., Tari, Z. (eds.) CoopIS 2002, DOA 2002, and ODBASE 2002. LNCS, vol. 2519, pp. 1317–1331. Springer, Heidelberg (2002)
14. Li, Y., Yang, H., Jagadish, H.V.: NaLIX: A generic natural language search environment for XML data. ACM Transactions on Database Systems (TODS) 32(4), 30 (2007)
15. Lincoln, M., Golani, M., Gal, A.: Machine-Assisted Design of Business Process Models Using Descriptor Space Analysis. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 128–144. Springer, Heidelberg (2010)
16. Lincoln, M., Karni, R., Wasser, A.: A Framework for Ontological Standardization of Business Process Content. In: International Conference on Enterprise Information Systems, pp. 257–263 (2007)
17. Markovic, I., Pereira, A.C., Stojanovic, N.: A framework for querying in business process modelling. In: Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI), Munchen, Germany (2008)
18. McGill, M.J., Salton, G.: Introduction to modern information retrieval. McGraw-Hill (1983)
19. Momotko, M., Subieta, K.: Process query language: A Way to Make Workflow Processes More Flexible. In: Benczúr, A.A., Demetrovics, J., Gottlob, G. (eds.) ADBIS 2004. LNCS, vol. 3255, pp. 306–321. Springer, Heidelberg (2004)
20. Shao, Q., Sun, P., Chen, Y.: WISE: a workflow information search engine. In: IEEE 25th International Conference on ICDE 2009, pp. 1491–1494. IEEE (2009)
21. van der Aalst, W.M.P., de Medeiros, A.K.A., Weijters, A.J.M.M.T.: Process Equivalence: Comparing Two Process Models Based on Observed Behavior. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 129–144. Springer, Heidelberg (2006)
22. van der Aalst, W.M.P., Ter Hofstede, A.H.M.: YAWL: yet another workflow language. Information Systems 30(4), 245–275 (2005)
23. van Dongen, B.F., Dijkman, R.M., Mendling, J.: Measuring Similarity Between Business Process Models. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 450–464. Springer, Heidelberg (2008)
24. Yan, Z., Dijkman, R., Grefen, P.: Business Process Model Repositories-Framework and Survey. Technical report, Beta Working Papers