Semantic Querying of Business Process Models

Ahmed Awad, Artem Polyvyanyy, and Mathias Weske
Business Process Technology Group
Hasso Plattner Institute at the University of Potsdam
D-14482 Potsdam, Germany
{Ahmed.Awad,Artem.Polyvyanyy,Mathias.Weske}@hpi.uni-potsdam.de

Abstract

Determining similarity between business process models has recently gained interest in the business process management community. So far similarity was addressed separately either at semantic or structural aspect of process models. Also, most of the contributions that measure similarity of process models assume an ideal case when process models are enriched with semantics—a description of meaning of process model elements. However, in real life this results in a heavy human effort consuming pre-processing phase which is often not feasible. In this paper we propose an automated approach for querying a business process model repository for structurally and semantically relevant models. Similar to the search on the Internet, a user formulates a BPMN-Q query and as a result receives a list of process models ordered by relevance to the query. We provide a business process model search engine implementation for evaluation of the proposed approach.

1 Introduction

In order to understand, communicate upon, or re-engineer working procedures, companies document their daily routines in the form of business process models. Business process modeling is a complex task. Model design consumes a considerable amount of time and requires determining of activities to be performed, ordering of their execution, handling exception cases that might occur, etc. Benefiting from the already developed process models seems to be a promising approach bound to reduce the time consumed to develop new models. A typical scenario where reuse and discovery of similar business processes is of interest is the case of companies merger. Both companies possess own business process descriptions. The companies want to facilitate integration task of operational processes and to minimize potential work overhead.

Techniques to detect similarity between process models have emerged as a way to assist the reuse of models [5, 11, 12, 16, 18]. In the mentioned approaches similarity has been addressed separately either at semantic or structural aspect of a process model. Also, most of the existing techniques require process models to be enriched with additional information: semantic activity description, explicit declaration of similar activities, etc. Usually, incorporation of such information is not foreseen by process modeling notations, like e.g., Business Process Modeling Notation (BPMN) [8], and is therefore left out in models. As a result, a manual step of supplying missing data is expected.

An automated business process model search technique is in demand. This paper proposes an approach for searching process models. It can be applied for the processes modeled using BPMN, and can be generalized to other modeling notations, e.g., Event-driven Process Chains (EPC) [15] or Workflow nets [1]. The presented approach assumes no process model notation extensions or additional information requirements.

We have developed BPMN-Q [3]—a language to query repositories of business process models. BPMN-Q allows expressing structural BPMN queries and specifies proceedings of determining whether a given process model is structurally similar to a query. In practice, the benefit of such an approach is limited by a human factor. People who model processes, as well as those who formulate queries, might use different vocabularies to express the same concepts. This leads to false similarity judgments. The problem can be addressed by applying Information Retrieval (IR) techniques. For instance, an analyst is interested in process models that describe the financial domain and handle loan applications such that eventually result in making an offer to a client. The analyst formulates a query as a request for process models containing the "*Offer*" activity. Further, an Information Retrieval technique can allow retrieving process models that contain the "*Propose*" activity as relevant to the query.

Discussed in this paper is a generic approach in terms

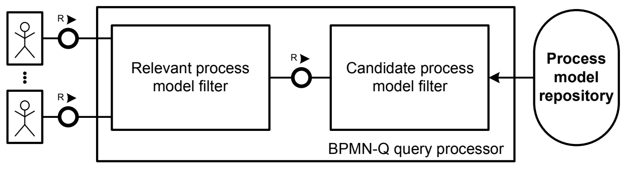


Figure 1. BPMN-Q component architecture

of usage of any Information Retrieval technique, e.g., Vector Space Model (VSM) [23] or Latent Semantic Indexing (LSI) [7, 10, 13]. However, enhanced Topic-based Vector Space Model (eTVSM) [17] is employed as a concrete solution. eTVSM allows capturing semantic similarities, encoded in eTVSM ontology, of natural language plain text documents and reflects them in document similarity values.

Contribution in this paper is a semantic expansion of BPMN-Q queries. Semantic expansion means employing an ontological dimension in the query matching process. In particular, ontology is used to tackle the problem of applying different terminologies when modeling processes. Ontology construction does not assume a priori semantic tagging or semantic description of process models. It is performed by employing the technique proposed and evaluated in [22] and uses only information available in models.

The rest of the paper is organized as follows. Section 2 provides background information on the approaches we build upon. A query expansion technique is then presented in section 3. A comprehensive example that demonstrates the result of applying a query expansion is given in section 4. Related work is discussed in section 5 before concluding the paper with a critical analysis of the approach and a look at future steps in section 6.

2 Foundations

The approach proposed in this paper is founded as a combination of two techniques: BPMN-Q and eTVSM. BPMN-Q is a language used to express structural process model queries, where as eTVSM is an Information Retrieval technique that allows semantic comparison of natural language plain text documents. Prior we start with a discussion of the paper contribution we would like to sketch a brief overview of these two techniques.

2.1 BPMN-Q

BPMN-Q [3] is a visual language based on BPMN. It is used to query business process models by matching a process model graph to a query graph. Figure 1 sketches a high level architecture overview of the *BPMN-Q query processor* component using FMC block diagram notation [14]. Users formulate process model structure related queries, BPMN-Q queries, and direct them to the component. The component filters out related process models from a process model repository by applying two internal filters. The "*Candidate process model filter*" performs initial preprocessing to reduce the search space. Afterwards, the "*Relevant process model filter*" completes the search by examining the candidates obtained on the previous step and filtering out the query related models only. A relevant model is triggered by discovering a query matching sub-graph within the examined process model.

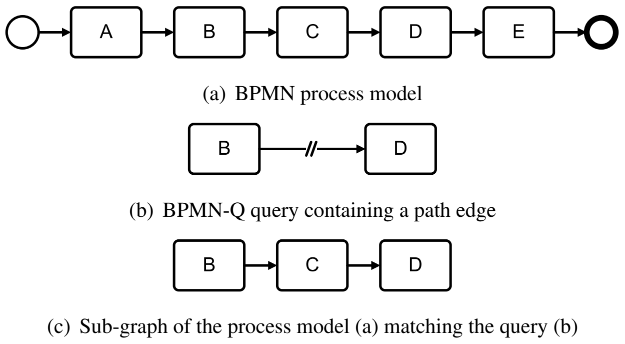

A
B
C
D
E
(a) BPMN process model
B
//
D
(b) BPMN-Q query containing a path edge
B
C
D
(c) Sub-graph of the process model (a) matching the query (b)

Figure 2. BPMN-Q query processing scenario

Figure 2 visualizes a BPMN-Q query processing scenario. In addition to the sequence flow edge in BPMN, BPMN-Q introduces the concept of a path (see Figure 2(b)). When matching a process graph (like the one in Figure 2(a)) to the query from Figure 2(b) (find models that have an execution path from activity B to activity D), the result is the sub-graph (see Figure 2(c)) composed of the query nodes and a path in between.

The expressive power of BPMN-Q as defined in [3] allows construction of more complex queries than just a path lookup. The language extends BPMN to include a variable activity, generic split/join gateways, and a generic node construct. Besides path edge, BPMN-Q introduces negative edge and negative path edge. In [4] BPMN-Q is used for the task of deadlock detection. In this paper, we limit ourselves to a subset of BPMN-Q that is sufficient to comprehend example scenarios shown later.

BPMN-Q query processing procedures are given using the formal definitions of a process graph and a query graph.

Definition 1 *A process graph is a tuple:*

$$PG = (\mathcal{N}, \mathcal{F}), \quad where:$$

- *$\mathcal{N}$—is a finite set of nodes which is composed of disjoint union of activities $\mathcal{N}_\mathcal{A}$, events $\mathcal{N}_\mathcal{E}$, and gateways $\mathcal{N}_\mathcal{G}$*

- $\mathcal{F} \subseteq \mathcal{N} \times \mathcal{N}$*—is a sequence flow relation between process graph nodes.*

Definition 2 *A query graph is a tuple:*

$$QG = (\mathcal{NQ}, \mathcal{FQ}, \mathcal{PQ}), \quad where:$$

- $\mathcal{NQ}$*—is a finite set of nodes which is composed of disjoint union of activities* $\mathcal{NQ}_{\mathcal{A}}$*, events* $\mathcal{NQ}_{\mathcal{E}}$*, and gateways* $\mathcal{NQ}_{\mathcal{G}}$

- $\mathcal{FQ} \subseteq \mathcal{NQ} \times \mathcal{NQ}$*—is a sequence flow relation between query graph nodes*

- $\mathcal{PQ} \subseteq \mathcal{NQ} \times \mathcal{NQ}$*—is a set of path edges between query graph nodes.*

BPMN-Q query processing is designed in a way that reduces the search space of process models to be checked. The procedure starts with identifying a set of process model candidates, rather than performing exhaustive query matching procedure on a complete repository. A process graph is a relevant candidate to a query graph only if the set of activity nodes in the process graph is a superset of the activity nodes in the query graph. Being a candidate process graph does not necessarily mean that it is relevant to the query (matches the query). A match to a query graph occurs when the structural relations (sequence flow, negative sequence flow, path, and/or negative path edges) between nodes in the query graph are satisfied by a process graph. The user can consult the log for candidate process models that are *not* relevant by viewing which structural relations were not satisfied by the process model.

BPMN-Q query matching procedure relies on the exact match of activity names in a query graph and activity names in a process graph.

2.2 eTVSM

Enhanced Topic-based Vector Space Model is a vector space model. Like Vector Space Model [23] or Topic-based Vector Space Model (TVSM) [6], eTVSM represents its concepts as vectors in a vector space. Relations between concepts are expressed through concept vector angles. In the case of eTVSM these angles express the level of semantic similarity between concepts. Semantic knowledge eTVSM is capable of operating with is represented in eTVSM ontology. eTVSM then proposes formal procedures for transforming ontology concepts to vectors in an operational vector space that map ontology semantic relations onto vector angles.

To construct an ontology, eTVSM uses concepts of *topics*, *interpretations*, and *terms*. These concepts are organized in a hierarchical, non-cyclic, directed graph structure.

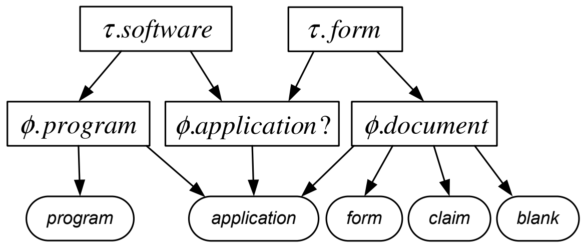


Figure 3. eTVSM ontology term to interpretation to topic assignment example

Edges of a graph aim at specifying concept semantic relations. A topic concept is the most general semantic entity of eTVSM ontology. Other ontology concepts refine existing topics. Inter-topic relations are expressed in a *topic map*. A topic map is a directed graph with topics as nodes. Graph edges assign super-, sub-topic relations. Interpretations are intermediate links between topics and terms. Conceptually, interpretations play role of semantic terms. An interpretation can be linked with an arbitrary number of topics. However, links between interpretations are not allowed. Terms are treated as the smallest information unit that has one or several semantic interpretations. To express this multiplicity in semantic meanings, a term might be linked with an arbitrary number of interpretations. Figure 3 shows the example of an extract from eTVSM ontology.

Proposed ontology extract consists of two topics τ*.software*, τ*.form* and three interpretations ϕ*.program*, ϕ*.application?*, and ϕ*.document*. Terms are leaf concepts in the ontology extract. Term *application* can have two clear interpretations of ϕ*.program* or ϕ*.document*. ϕ*.application?* interpretation is designed for cases when it is not possible to derive a clear distinction in the term meaning.

eTVSM is used in Information Retrieval to obtain similarity level of two natural language documents [17, 22]. Prior to the text comparison procedure, models for target documents are constructed. A document model that represents the document $d_j \in D$ (D—is a set of all documents) is represented by a document vector and is defined as:

$$\forall d_j \in D : \vec{d_j} = \frac{1}{|\vec{\delta_j}|}\vec{\delta_j} \quad \Rightarrow \quad \left|\vec{d_j}\right| = 1$$
$$\text{with} \quad \vec{\delta_j} = \sum_{\phi_i \in \Phi} \omega_{d_j,\phi_i}\vec{\phi_i}.$$

A document vector is a weighted sum of interpretation vectors included in the document model. Here, ω_{d_j,ϕ_i} is the weight of the interpretation ϕ_i ($\phi_i \in \Phi$—is a set of all interpretations) in the document d_j, e.g., a simple concept occurrence number within a document. Discussion on a formal procedure for obtaining interpretation vectors is omitted in this paper, but can be found in [17, 22]. Basically, interpretation vector construction heuristics considers eTVSM ontology graph structure to express interpretation similarity level as a vector angle and is based on prior construction of topic concept vectors. As long as only vector direction is relevant for obtaining angles between vectors, document vectors are normalized to the length of one. A document vector length is obtained as:

$$\begin{aligned}
\left|\vec{\delta}_i\right| &= \left|\sum_{\phi_k \in \Phi} \omega_{d_i,\phi_k} \vec{\phi}_k\right| = \sqrt{\left|\sum_{\phi_k \in \Phi} \omega_{d_i,\phi_k} \vec{\phi}_k\right|^2} \\
&= \sqrt{\left(\sum_{\phi_k \in \Phi} \omega_{d_i,\phi_k} \vec{\phi}_k\right)^2} \\
&= \sqrt{\sum_{\phi_k \in \Phi} \sum_{\phi_l \in \Phi} \omega_{d_i,\phi_k} \omega_{d_i,\phi_l} \vec{\phi}_k \vec{\phi}_l}
\end{aligned}$$

Finally, the similarity level of two documents d_i and d_j is obtained as the scalar product of corresponding document vectors (see Definition 3). Considering the document vector normalization, the similarity value becomes equal to the cosine of the angle between document vectors.

Definition 3

$$\begin{aligned}
sim(d_i, d_j) &= \vec{d}_i \vec{d}_j = \frac{1}{\left|\vec{\delta}_i\right|} \vec{\delta}_i \frac{1}{\left|\vec{\delta}_j\right|} \vec{\delta}_j = \frac{1}{\left|\vec{\delta}_i\right| \left|\vec{\delta}_j\right|} \vec{\delta}_i \vec{\delta}_j \\
&= \frac{1}{\left|\vec{\delta}_i\right| \left|\vec{\delta}_j\right|} \sum_{\phi_k \in \Phi} \omega_{d_i,\phi_k} \vec{\phi}_k \sum_{\phi_l \in \Phi} \omega_{d_j,\phi_l} \vec{\phi}_l \\
&= \frac{1}{\left|\vec{\delta}_i\right| \left|\vec{\delta}_j\right|} \sum_{\phi_k \in \Phi} \sum_{\phi_l \in \Phi} \omega_{d_i,\phi_k} \omega_{d_j,\phi_l} \vec{\phi}_k \vec{\phi}_l
\end{aligned}$$

Such a document similarity value of eTVSM Information Retrieval model assumes semantic relations between terms employed in the compared documents. Semantic knowledge is represented in a prior constructed eTVSM ontology. eTVSM is an advanced Information Retrieval model that can represent most of the linguistic phenomena encoded into eTVSM ontology: inflection, composition, derivation, synonymy, hyponymy, meronymy, homography, metonymy, and word groups [17, 22].

3 BPMN-Q Query Expansion

In Information Retrieval a query expansion is a process of reformulating a seed query to improve effectiveness of search results. In the case of Web search engines a query expansion process involves evaluation of a user query input and expanding the query text to match additional documents. A query expansion process might involve manual or automatic techniques including thesaurus lookup or studying of user action logs [9]. In this paper we would like to make a shift from a natural language query expansion to a process structure query (BPMN-Q query) expansion. As a requirement, BPMN-Q query expansion should be performed in an automated way.

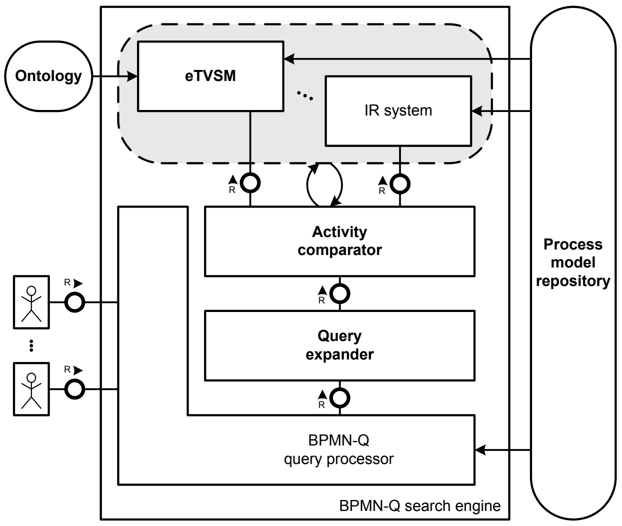


Figure 4. BPMN-Q search engine architecture

In the core of the approach for BPMN-Q query expansion lies substitution of activities of a query graph (see Definition 2) with similar. The method of obtaining activity similarities bases on the similarity level of activities treated as natural language documents. Process model activities are usually named and sometimes supplied with detailed textual descriptions. One might apply well-known Information Retrieval techniques, like VSM or LSI, to derive similarities of activity names or their textual descriptions. The heuristic is then to accept such values as a measure of activities similarity. In the work reported the function of obtaining plain text similarities is entitled to eTVSM.

BPMN-Q expansion-enabled interface will take as input a given query, look for similar activities to ones employed in the query, construct new queries based on found activity alternatives, and return results that match either the seed query or any of its derivatives.

In Figure 4 the architecture of a BPMN-Q search engine, modeled using FMC block diagram notation, is provided. The architecture incorporates the query expansion interface. The engine is triggered for action by user requests passing BPMN-Q process model queries to the *BPMN-Q query processor* component. The *BPMN-Q query proces-*

sor is responsible for performing process model search in the process model repository for a given query and arranging search results for a set of expanded queries prior delivering them to a user. The *BPMN-Q query processor* requests a list of expanded queries by passing a seed query to the *Query expander* component of the engine. To perform its task the *Query expander* component asks the *Activity comparator* component for the similar activities to those employed in the seed query. The *Activity comparator* component can rely in its work on any Information Retrieval system. In the case when eTVSM is used, an eTVSM specific ontology is required. For each specific activity the component is able to derive a list of semantically similar activities. Obtained eTVSM similarity values exploit semantic knowledge of prior constructed eTVSM ontology. One might further specify a similarity level threshold for accepting activities as similar. Afterwards, the original query gets modified by substituting any of its activities with similar ones. With such a substitution step, a new query graph is constructed—an expanded BPMN-Q query. At the end, the final search results are obtained from the sets of results of the seed as well as all of the expanded queries. The order on retrieved process models can be derived based on the similarity level of employed expanded queries and the seed query.

Discussion of different scopes of activity similarities and their effect on the quality of search results is given in section 3.1. In section 3.2 an approach of constructing eTVSM ontology based on a given set of process models is presented. Finally, the formulae upon which an ordering of the final search results is obtained constitutes section 3.3.

3.1 Scopes of Activity Similarities

The architecture solution provided in Figure 4 includes a building block representing a component for obtaining activity similarities—the *Activity comparator* component. It is designed to deliver similarity level of any pair of activities met in a process model repository. The solution is proposed at the activity model level. Activity model describes a set of similar activity instances and can be expressed in different forms, e.g., by plain text (activity name or detailed textual description), by some formal specification, or by references to software components that implement it [24].

Activity models represent the $M1$ layer of Meta Object Facility (MOF) [21], while activity instances correspond to the $M0$ layer. Figure 5 shows the relationship between activity models and activity instances as well as visualizes different ways to express activity models. We would like to get use from the fact that activity models can be represented in the form of plain text and apply Information Retrieval techniques to derive plain text similarities of activity model descriptions. Obtained similarity values are then accepted as activity similarity values. Therefore, the proposed ap-

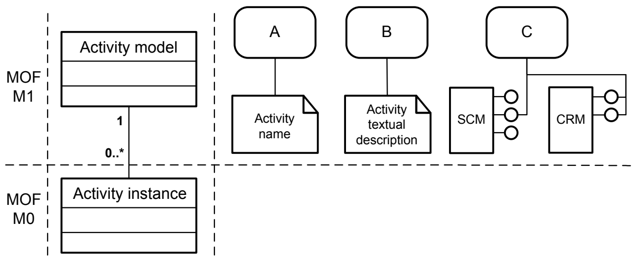


Figure 5. MOF: Activity models and activity instances

proach of activity comparison heavily relies on the modeling technique that assumes specification of activity models by names or by textual descriptions.

In the case when operating with activity model names, activity model "*Check loan application*" can be questioned upon similarity level with activity models of the following names: "*Loan application check*", "*Examine loan application*", "*Check loan documentation*", "*examine loan documentation*", "*go over loan application*". Intuitively, by applying the proposed approach of textual comparison one should expect a high similarity value for any pair of listed activity models.

In the case when activity models are supplied with textual semantic descriptions of the underlying activities, the approach acts similar to the situation with activity names. The only difference is that instead of activity names, textual descriptions are compared. Detailed activities descriptions provide more information for correct similarity judgments.

In this paper we narrow the approach down to the case when activity models are specified by names. It is a common modeling practice when activity models are just named. This is explained by an attempt to keep modeling effort low. Thus, the approach targets the most general case of business process modeling technique and allows extension in the case when activity models are specified by detailed textual descriptions.

3.2 eTVSM Ontology Construction

The developed approach is generic in terms of usage of any Information Retrieval technique that delivers plain text document similarities. eTVSM is chosen as a concrete option. It is intended to reflect semantic similarity measure of activities in obtained activity model similarity values, and eTVSM completes the approach to allow this. By relying on ontology, eTVSM is able to exploit semantic document similarities.

Prior that eTVSM can start with obtaining document similarity values and aid in expanding BPMN-Q queries,

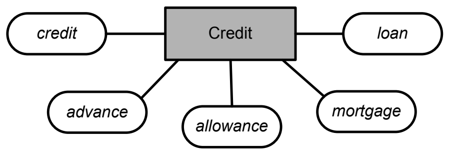


Figure 6. eTVSM ontology total synonymy modeling pattern example

ontology needs to be constructed. eTVSM ontology is the semantic knowledge base we aim to reuse. The quality of eTVSM similarity values greatly depends on eTVSM ontology structure. A completely automated "synonymy eTVSM" ontology construction approach proposed and evaluated in [22] is employed. Synonymy is the fact that many equivalent or closely related meanings can be conveyed by distinct words. "Synonymy eTVSM" automatically extracts and reuses synonymy knowledge of WordNet[1]. The example of eTVSM ontology total synonymy modeling pattern is presented in Figure 6. Here, eTVSM ontology modeling notation discussed in [17, 22] is used. A shaded rectangle represents interconnected topic and interpretation with the same name. Terms are depicted as ellipses.

What is given as input for eTVSM ontology construction is a business process model repository. On the first step a list of all activity models present in the repository is obtained. Afterwards, for each recognized term from activity model names the corresponding synonymy pattern (see Figure 6) from WordNet is imported into eTVSM ontology. The WordNet concept that matches to the synonymy pattern is a *synset* [22]. If there are multiple synsets for the same term present in WordNet, then the most commonly used one is selected. Specific details about the "synonymy eTVSM" ontology construction approach can be found in [22].

3.3 Order on Queries

The mechanism of obtaining similar activities allows us to construct expanded BPMN-Q queries by substituting activities from a seed query by similar ones. In such a way one might derive a set of queries that were obtained due to the expansion of the seed query. Each of such expanded queries can be then questioned for relevant process models from a repository. Result of a single repository search is now a union of all the results of the seed and the expanded queries. How these results can be organized and presented to a user? This is the question addressed in this section. We propose to first derive an order on expanded queries and then to transfer this order on results of the queries.

[1]Semantic lexicon for the English language. It groups English words into sets of synonyms called synsets.

For each activity model $a_i \in \mathcal{NQ}_{\mathcal{A}s}$ from a seed BPMN-Q query $Q_s = (\mathcal{NQ}_{\mathcal{A}s} \cup \mathcal{NQ}_{\mathcal{E}s} \cup \mathcal{NQ}_{\mathcal{G}s}, \mathcal{FQ}_s, \mathcal{PQ}_s)$ one might construct a set $S_{i,n} = \{s \,|\, sim(a_i, s) \geq n\}$ of similar activity models s from a process model repository that show similarity level with a_i equal or higher than n. A threshold value n can be adjusted to a desired level separately for each activity model, e.g., as a feature of a tool that supports process queries. Under the assumption that the similarity level can be seen as the probability of activity models to be equal (to model the same activity) and that structure of both queries is the same, the probability of two BPMN-Q queries, i.e., the seed query Q_s and the expanded query Q_e to be similar, can be given as:

$$SIM(Q_s, Q_e) = \prod_{i=1}^{k} sim(a_i, s_i),$$

where:

- $s_i \in S_{i,n} \cap \mathcal{NQ}_{\mathcal{A}e}$
- $k = |\mathcal{NQ}_{\mathcal{A}s}|$
- $a_i \in \mathcal{NQ}_{\mathcal{A}s}$.

An order on BPMN-Q queries can be obtained from the order on similarities of the expanded queries to the seed query. Afterwards, the results of the seed query search can be ordered according to the order on the expanded queries. First, the results of the seed query are displayed, next the results of the most similar expanded query, and so on until the results of the least similar query are presented.

4 Example Scenario

In this section we demonstrate the results of the work of the implemented business process model search engine. The work of the engine follows the principles presented in this paper. An example scenario starts with a business analyst having a demand for a process model representing a client loan application handling. The scenario consists of formulating a search query (accomplished by the analyst), consequently performing a search for the relevant process models from a process model repository for the given query (accomplished by the implemented process model search engine).

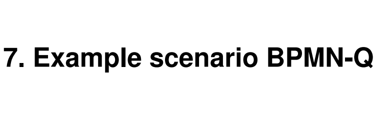


Figure 7. Example scenario BPMN-Q query

Order	Activity	Similarity
1	Check loans	0.816
2	Check completeness of loan application	0.774
3	Check credit	0.408
4	Check client assessment	0.333
5	Revise fulfillment of loan request	0.204

Table 1. List of activities similar to the "Check loan application" activity

Order	Activity	Similarity
1	Print offers	0.707
2	Produce offer	0.707
3	Offer further products	0.577

Table 2. List of activities similar to the "Offer" activity

Order	Act. A	Act. B	Threshold	Similarity
1	1	1	0.7	0.577
2	2	1	0.7	0.547
3	2	2	0.7	0.547
4	3	1	0.4	0.288
5	4	3	0.2	0.192
6	5	1	0.2	0.144

Table 3. List of expanded queries derived from query visualized in Figure 7

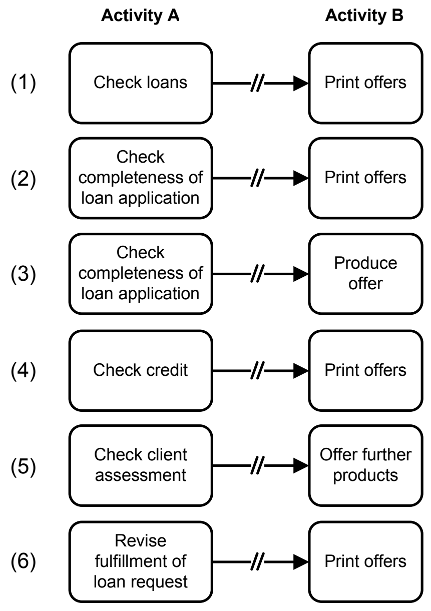


Figure 8. Expanded queries from Table 3

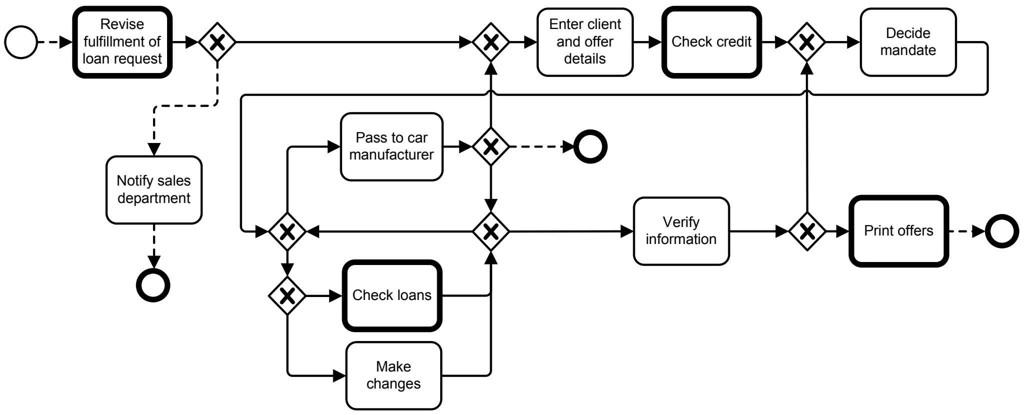
The BPMN-Q query employed for the described example scenario is visualized in Figure 7.

The analyst formulates his/her search intent as a lookup for models that possess the "*Check loan application*" activity or similar, and have an execution path from it to the "*Offer*" activity or similar.

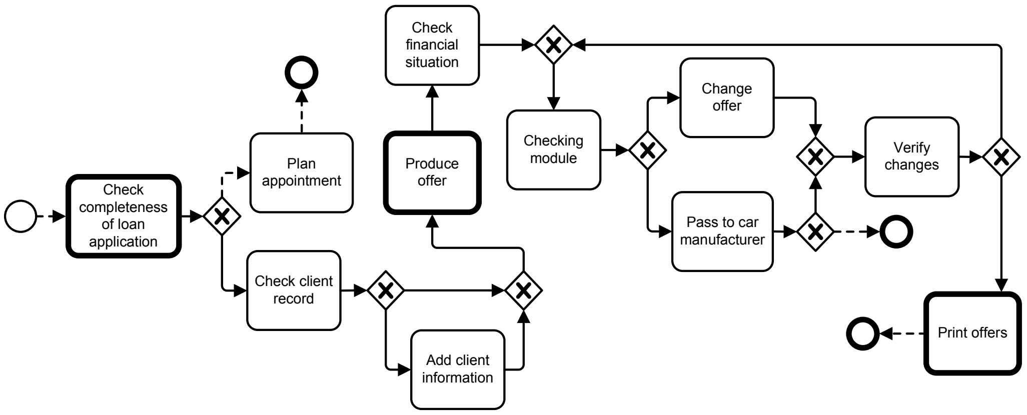
Starting with the activity similarity threshold of 1.0 for each activity in the seed query (an exact query match) results in no process models found in the repository to be relevant to the query. Therefore, we take the strategy of stepwise decreasing of the similarity threshold for both activities from the seed query. This way we increase the search space. Table 1 provides an ordered by similarity to the "*Check loan application*" activity list of activities found in the studied process model repository. Respectively, Table 2 provides a list of activities similar to the "*Offer*" activity.

All possible expanded queries composed of similar activities retrieved 3 models from the studied process model repository. These models are presented in Figure 9. They are relevant to the 6 expanded queries listed in Table 3. The query with order number 1 is obtained by substituting activities in the seed query: the "*Check loans*" activity (order number 1 in Table 1—column "Act. A" in Table 3) instead of the "*Check loan application*" activity and the "*Print offers*" activity (order number 1 in Table 2—column "Act. B" in Table 3) instead of the "*Offer*" activity. The resulting expanded query formulates a search intent for process models that contain the "*Check loans*" activity and have an execution path from it to the "*Print offers*" activity. The expanded query was obtained after setting the activity similarity threshold to 0.7 and has a similarity value with the seed query equal to $0.816 \cdot 0.707 \cong 0.577$.

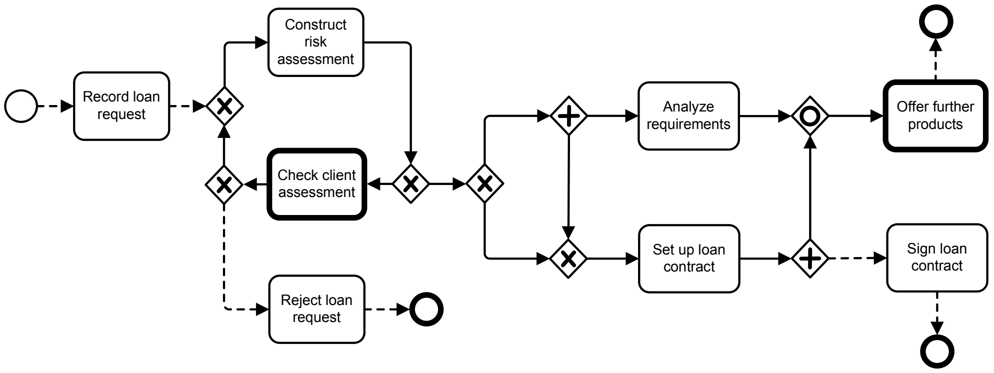
The process model from Figure 9(a) was retrieved as relevant for the queries with order numbers 1, 4 and 6 in Table 3. The process model from Figure 9(b) was judged as relevant for the queries 2 and 3 in Table 3. Finally, the process model from Figure 9(c) is relevant to the query with order number 5 in Table 3. As you have probably noticed, one process model can be retrieved as relevant to more than one expanded query. In the case of process model from Figure 9(a), it is relevant to 3 expanded queries. However, the model should be included only once into the final results as relevant to the most similar expanded query to the seed query, i.e., the query with order number 1 in Table 3. Taking into consideration the rationale discussed in section 3.3, the results of the example scenario query search should be displayed in the order: first process model from Figure 9(a), then 9(b), and finally 9(c). Visualized as solid control flow edges in process models from Figure 9 represent matching sub-graphs superimposed for each expanded query the model is relevant to. The dashed edges represent the control flow not on the path relevant for the process structure matching procedure. The activities from the expanded queries are



(a) Process model matching queries 1, 4 and 6 from Table 3



(b) Process model matching queries 2 and 3 from Table 3



(c) Process model matching query 5 from Table 3

Figure 9. Search results for the BMPN-Q query from Figure 7 (activities from the expanded queries are emphasized with a bold borderline, while solid edges represent the control flow on the query matching path). Provided models are adopted from [11, 20]

emphasized with a bold borderline in Figure 9.

As the search result the business analyst obtains 3 process models as relevant to the seed query instead of no results if applying a pure BPMN-Q search procedure. The chances of retrieving desired model in less search iterations are increased.

5 Related Work

The topic of obtaining process model similarities has gained a lot of attention recently. The approach to a decision on two process models to be similar has been addressed from different perspectives. Process variants [18] is an approach to derive similarity by deciding one model to be a specific variant of another by applying model reductions. Process variants are also discussed in [16], but based on linguistic similarities between activity descriptions in both business process models. The approach is similar to the one presented in this paper considering the usage of activity semantic similarity in deciding on process models similarity. However, the approach does not consider the structural similarity of both models.

A method for measuring similarities between business processes modeled in Web Ontology Language (OWL) [2] is discussed in [12]. One of the goals of the work is to come up with a similarity measure between two process models, e.g., a process reference model and an existing detailed process model. The approach assumes business process models to be described in an unambiguous format which enables automated reasoning. In the case this was not foreseen beforehand, the approach assumes the overhead of semantic OWL process model description. In the approach discussed in this paper we allow process modelers to apply their term vocabularies. The Information Retrieval techniques are then designed to solve the problem of ambiguous term interpretations or vocabulary variants. Semantic process similarity judgments are carried out by means of semantic aware Information Retrieval methods, like eTVSM.

In [11], a classification of similarities and differences between business processes is discussed. The author defines possible differences between processes that are assumed to be similar. As the first step a process analyst determines semantic equivalence between sets of activities, as well as equivalent roles, in the studied process models. Afterwards, the approach can detect differences between processes. This approach addresses a situation of binary comparison between process models and includes a manual step of semantic pairing of process activities.

An approach that describes similarity between processes based on the structural measure is proposed in [5]. Two processes are compared by means of calculating a so-called dependency distance measure which reflects the difference in the underlying process control flow graphs. The study assumes the same set of activities in compared process models. The semantic process models similarity measurement is not addressed.

In [19] the authors discuss behavioral similarity of business process models. Based on the process model control flow graph a causality graph footprint of a process is derived. Afterwards, it is proposed to construct footprint vectors and to derive similarity, like in VSM or eTVSM, as a cosine of the angle between two footprint vectors. The approach requires that similar activity models are exactly the same in compared process models, i.e., have the same names. Otherwise, the approach requires manual intrusion.

6 Conclusions

In this paper we have introduced the expansion of BPMN-Q queries. BPMN-Q allows users to formulate structure related process model queries and to retrieve matching models from a process model repository. By applying Information Retrieval techniques we were able to increase the seed BPMN-Q query search space. Moreover, by using eTVSM that exploits WordNet knowledge the search process evolves to become semantic aware. Theoretical sentiment obtained empirical verification in the implemented business process model search engine capable of running BPMN-Q queries. Operational results of the developed tool were presented and discussed in the paper. Expansion of a query search space is controlled by the similarity threshold that can be adjusted to deliver an exact query match or allow an exhaustive search over a process model repository. The similarity threshold does not control the structural similarity between a query graph and a process graph. As pointed in section 2.1 matching is exact. So, if the user is not sure about the order in which activities appear in the process model, he/she can simply relax these ordering constraints. From a theoretical point of view, a similarity threshold of value 0 will allow the exhaustive search of all the repository. On the implementation level, this might lead to performance problems specially in cases of queries having a large set of activity nodes. The performance bottleneck stems from the fact that the generation of expanded queries is non-polynomial.

Presented approach allows an automated operation from start under the condition that activity model names are provided in the form of plain text—a reasonable assumption. Further hints for the overall approach improvement in the case if activity models are enhanced by detailed textual descriptions are discussed.

In the discussed scenario of companies merger, the presented approach aims at automating the discovery of similar processes in order to ease the task of integrating them. Several companies have to integrate their process model repositories to be able to run queries and discover process similarities. Moreover, the proposed approach is helpful to the company afterwards in getting fast overview of the company's process ecosystem, e.g., prior of introducing a new process.

As the future enhancement steps of the approach we foresee introduction of new process model search criteria with their integration into the implemented search engine. Currently, the evaluation of the approach was performed for processes modeled in BPMN; further generalization of the methodology for other process modeling notations is a future work. Finally, current evaluation results were obtained after running queries over a process model repository composed of about 100 models. Operational evaluation of the search engine for larger repositories is a future work of the highest priority.

References

[1] W. Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[2] G. Antoniou and F. van Harmelen. Web ontology language: OWL, 2003.

[3] A. Awad. BPMN-Q: A Language to Query Business Processes. In *EMISA*, pages 115–128, 2007.

[4] A. Awad and F. Puhlmann. Structural detection of deadlocks in business process models. In *BIS*, pages 239–250, 2008.

[5] J. Bae, L. Liu, J. Caverlee, and W. B. Rouse. Process Mining, Discovery, and Integration using Distance Measures. In *ICWS '06: Proceedings of the IEEE International Conference on Web Services (ICWS'06)*, pages 479–488, 2006.

[6] J. Becker and D. Kuropka. Topic-based Vector Space Model. In *Proceedings of the 6th International Conference on Business Information Systems*, pages 7–12, 2003.

[7] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using Linear Algebra for Intelligent Information Retrieval. Technical Report UT-CS-94-270, 1994.

[8] BPMI.org. *Business Process Modeling Notation*, 1.0 edition, May 2004.

[9] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Query Expansion by Mining User Logs. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):829–839, 2003.

[10] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[11] R. M. Dijkman. A Classification of Differences between Similar Business Processes. In *EDOC*, pages 37–50. IEEE Computer Society, 2007.

[12] M. Ehrig, A. Koschmider, and A. Oberweis. Measuring similarity between semantic business process models. In *APCCM '07: Proceedings of the 4th Asia-Pacific Conference on Conceptual Modelling*, pages 71–80, 2007.

[13] T. Hofmann. Probabilistic Latent Semantic Analysis. In *Proceedings of Uncertainty in Artificial Intelligence, UAI'99*, 1999.

[14] F. Keller and S. Wendt. FMC: an approach towards architecture-centric system development. In *Proceedings of the 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems*, pages 173–182, 2003.

[15] G. Keller, M. Nüttgens, and A. Scheer. Semantische Prozessmodellierung auf der Grundlage "Ereignisgesteuerter Prozessketten (EPK)". Technical Report Heft 89 (in German), Veröffentlichungen des Instituts für Wirtschaftsinformatik University of Saarland, 1992.

[16] A. Koschmider and A. Oberweis. How to detect semantic business process model variants? In *SAC*, pages 1263–1264, 2007.

[17] D. Kuropka. *Modelle zur Repräsentation natürlichsprachlicher Dokumente*. Logos Verlag, 2003.

[18] R. Lu and S. W. Sadiq. Managing Process Variants as an Information Resource. In *Business Process Management*, pages 426–431, 2006.

[19] J. Mendling, B. Dongen, and W. Aalst. On the Degree of Behavioral Similarity between Business Process Models. In *CEUR-Workshop*, pages 39–58, 2007.

[20] M. Nüttgens and F. J. Rump. Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In *Promise*, 2002.

[21] Object Management Group, Inc. *Meta Object Facility (MOF) 2.0 Core Specification*, oct 2003.

[22] A. Polyvyanyy and D. Kuropka. A Quantitative Evaluation of the Enhanced Topic-Based Vector Space Model. Technical Report 19, Hasso Plattner Institute, 2007.

[23] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[24] M. Weske. *Business Process Management*. Springer, 2007.