Software & Systems Modeling (2019) 18:1179–1194
https://doi.org/10.1007/s10270-017-0649-y

SPECIAL SECTION PAPER

Searching textual and model-based process descriptions based on a unified data format

Henrik Leopold[1] · Han van der Aa[1] · Fabian Pittke[2] · Manuel Raffel[2] · Jan Mendling[2] · Hajo A. Reijers[1]

Received: 16 December 2016 / Revised: 26 November 2017 / Accepted: 29 November 2017 / Published online: 22 December 2017

Abstract
Documenting business processes using process models is common practice in many organizations. However, not all process information is best captured in process models. Hence, many organizations complement these models with textual descriptions that specify additional details. The problem with this supplementary use of textual descriptions is that existing techniques for automatically searching process repositories are limited to process models. They are not capable of taking the information from textual descriptions into account and, therefore, provide incomplete search results. In this paper, we address this problem and propose a technique that is capable of searching textual as well as model-based process descriptions. It automatically extracts activity-related and behavioral information from both descriptions types and stores it in a unified data format. An evaluation with a large Austrian bank demonstrates that the additional consideration of textual descriptions allows us to identify more relevant processes from a repository.

Keywords Integrated Process Search · Unified Data Format · Textual Process Descriptions

1 Introduction

Business process models have proven to be effective means for the visualization and improvement in complex organizational operations [11]. However, not all process-related information is available in the form of process models. On the one hand, because the creation of process models is a time-consuming endeavor that requires considerable resources [24]. On the other hand, because not all process information is best captured as a process model [48]. In particular, *work instructions* that describe tasks at a high level of detail are often documented in the form of textual descriptions, as this format is more suitable for specifying a high number of details [4]. As a result, process repositories in practice do not only consist of process models, but often also contain textual process descriptions. These are linked to individual activities of the process models in order to specify the detailed action items behind them.

The problem of this supplementary use of textual descriptions in process repositories is that automatic analysis techniques designed for process models, such as weakness identification [5], service identification [32], or compliance queries [2], might provide incomplete results. Suppose a company aims to increase the share of digital communication; then, it can query its process repository to find all processes that still include paper-based communication. The query results, however, will be limited to the process models that indicate the use of paper-based communication already in their activity text labels. Process models that describe the process at a higher level of abstraction, but link to textual descriptions revealing that this process is indeed associated with paper-based communication, will be ignored. To the best

Communicated by Dr. Ilia Bider and Dr. Rainer Schmidt.

✉ Henrik Leopold
h.leopold@vu.nl

Han van der Aa
j.h.vander.aa@vu.nl

Fabian Pittke
fabian.pittke@wu.ac.at

Manuel Raffel
manuel.raffel@wu.ac.at

Jan Mendling
jan.mendling@wu.ac.at

Hajo A. Reijers
h.a.reijers@vu.nl

[1] Vrije Universiteit Amsterdam, De Boelelaan 1081, HV 1081 Amsterdam, The Netherlands

[2] WU Vienna, Welthandelsplatz 1, 1020 Vienna, Austria

of our knowledge, there is currently no technique available that provides the possibility to search textual and model-based process descriptions in an integrated fashion. One explanation for the absence of such a technique might be the challenges that are associated with it. Among others, it requires the definition of an integrated data format that is able to represent both textual and model-based process descriptions.

Against this background, we use this paper to follow up on our earlier work [30] and to propose a technique that can search both text and model-based process descriptions. It combines natural language analysis techniques in a novel way and transforms textual as well as model-based process descriptions into a unified data format. By integrating technology from the semantic web domain, we facilitate the possibility of performing comprehensive search operations on this data format. To define this technique, we follow the Design Science Research Methodology [23,46] as specified by [42]. It consists of six core activities: (1) problem identification and motivation, (2) definition of the objectives of the design artifact, (3) design and development of the design artifact, (4) demonstration of the problem-solving efficacy of the design artifact, (5) measurement of how well the design artifact supports the solution to the problem, and (6) communication of the design artifact to researchers and other relevant audiences.

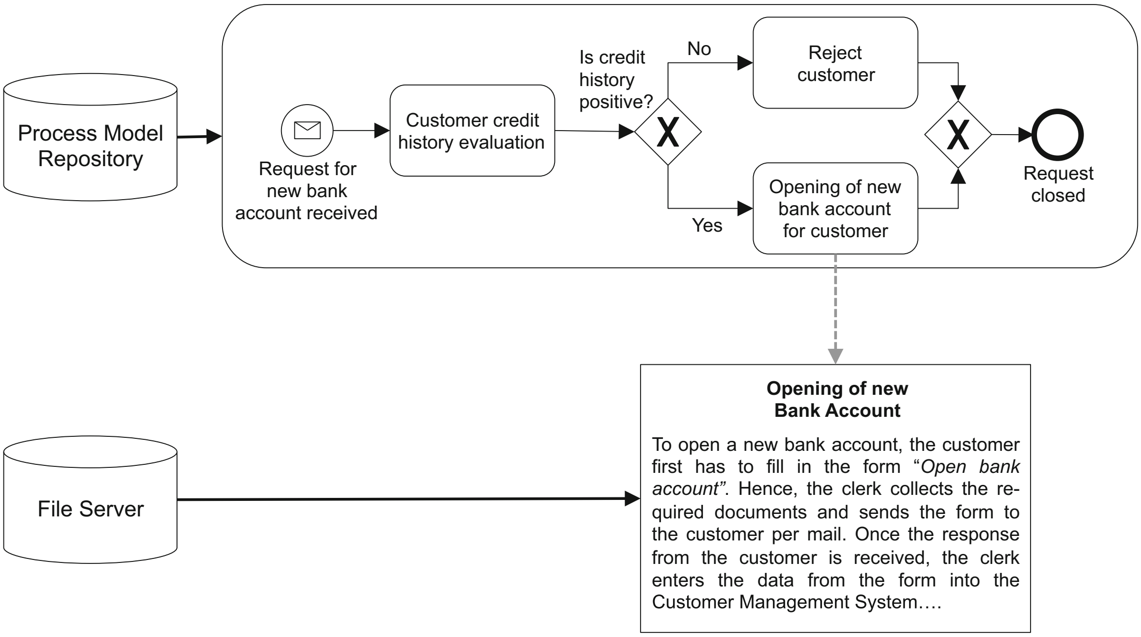
By implementing these steps, the paper is organized as follows. Section 2 introduces the problem of searching textual and model-based process descriptions and discusses related work. Section 3 then introduces our proposed technique on a conceptual level. Section 4 presents the results of an evaluation with a large Austrian bank. Section 5 concludes the paper and provides an outlook on future research.

2 Background

This section introduces the background of our research. First, we illustrate the problem of searching textual and model-based process descriptions. Then, we reflect on related techniques that are currently available.

2.1 Motivating example

In order to illustrate the importance of textual descriptions in the context of a process search, let us take a look at the implications of only taking process models into account. To this end, consider the example shown in Fig. 1. It shows a simple process model created using the Business Process Modeling and Notation (BPMN) and a small complementary text from a bank. The process model is part of a process model repository; the complementary text is stored on a separate file server. We can see that the business process is triggered by the request to open a new bank account. Subsequently, the credit history of the customer is evaluated. The outcome of this evaluation can be either positive or negative. In case of a negative credit evaluation, the customer is rejected. If the credit history evaluated as positive, a new bank account is opened. Finally, the request is closed. In addition to the BPMN process model, there is complementary text. It further


Process Model Repository
Request for new bank account received
Customer credit history evaluation
Is credit history positive?
No
Yes
X
Reject customer
Opening of new bank account for customer
X
Request closed
File Server

Opening of new Bank Account

To open a new bank account, the customer first has to fill in the form "*Open bank account*". Hence, the clerk collects the required documents and sends the form to the customer per mail. Once the response from the customer is received, the clerk enters the data from the form into the Customer Management System.…

Fig. 1 Exemplary process model with complementary natural language description

specifies the details of the activity *"Opening of new bank account for customer"*. Among others, it describes that the opening of a bank account is associated with a mail-based information exchange with the customer.

Assume this process model is part of the process repository of an organization. If this organization was interested in all business processes that involve an interaction with a customer, automated search techniques would have no difficulties to identify the depicted process. That is the case because the customer is explicitly mentioned in the activity labels, e.g., in *"Customer credit history evaluation"*. However, suppose the organization aims at improving its operations by replacing all mail-based correspondence with an electronic alternative. In this case, an automated search on the process model would not identify any potential for improvement. That is because the activities of the process model do not contain any words that might be associated with the activities of mailing or sending. Only the description attached to the activity *"Opening of new bank account"* explicitly refers to sending a form per mail.

This example illustrates the advantage of performing search operations that cover textual as well model-based process descriptions. As a respective technique is currently missing, it is our goal to define such a technique in this paper.

2.2 Related work

The work from this paper relates to three major streams of research: techniques for process model search, techniques for semantic process model analysis, and techniques for the analysis of process behavior in textual resources.

Techniques for *process model search* can be divided into two main groups. The first group consists of techniques focusing on *structure*. They compare query and process model with respect to behavioral properties, for instance, whether two activities occur in a particular order. Among others, such structural querying techniques have been defined based on temporal logical [1], the weak ordering formalism [28], and indexing [25,54]. The limitation of these structural techniques is that they typically assume that semantically identical activities have identical or similar labels. The second group of search techniques focuses on the *content from text labels* and tries to overcome this problem. Among others, they employ NLP techniques to identify similar models based on their activity labels. Notable examples for such techniques have been defined by Awad et al. [2] and Qiao et al. [45], where the authors use dictionaries and language modeling to retrieve semantically similar models.

Techniques for *semantic process model analysis* aim to ensure that the activity labels of process models are correctly and consistently defined. A major challenge in this context is the accurate detection of the label structure, i.e., which word of the label represents the verb and which word represents the object. Techniques addressing this problem have been defined by Becker et al. [6] and Leopold et al. [31]. Other works address the problem of linguistic variations in process model labels. Koschmider and Blanchard [27] and also Breuker et al. [7] provide approaches for consistently specifying process elements in the context of a process hierarchy. Van der Vos et al. [51] use a semantic lexicon to check whether the terms used in a model are sufficiently meaningful. Pittke et al. [43] take a more general perspective and, among others, check whether process models contain misleading synonyms. Based on such semantic considerations, also techniques for detecting errors in process models have been defined. The technique by Gruhn and Laue [21], for instance, recognizes semantically impossible behavior of process models.

Techniques that *analyze process behavior in textual resources* aim to recognize and extract behavioral relations between activities by employing natural language analysis tools. Most existing techniques perform this task for the purpose of generating a model-based description from the analyzed text. There are techniques that generate BPMN process models from textual process descriptions [17], group stories [19], use cases [47], and heterogeneous textual sources [18]. The main challenge that these techniques must overcome is the ambiguity of natural language. Words and sentences can have various meanings and, therefore, may allow for several interpretations of the process behavior. A recent technique by Van der Aa et al. [49] tries to address this ambiguity in a comprehensive way by deriving a so-called behavioral space from a text. This behavioral space captures all possible process interpretations of a text in a structured manner.

Despite the considerable number of techniques for process model search and analysis, a conceptual solution for an integrated search technique is still missing. To develop such a technique, we need to define a data format that allows us to store the information extracted from both process description types in a unified way. Based on such a format, we can then perform search operations covering model-based as well as textual process descriptions.

3 Conceptual approach

In this section, we introduce our approach for comprehensive process search by integrating textual and model-based process descriptions. Section 3.1 gives an overview of the architecture of our approach. Section 3.2 describes the unified format we use to integrate textual and model-based content. Sections 3.3 and 3.4 show how we parse and transform textual and model-based descriptions into the unified format. Finally, Sect. 3.5 illustrates how we use the unified data format for integrated process search.

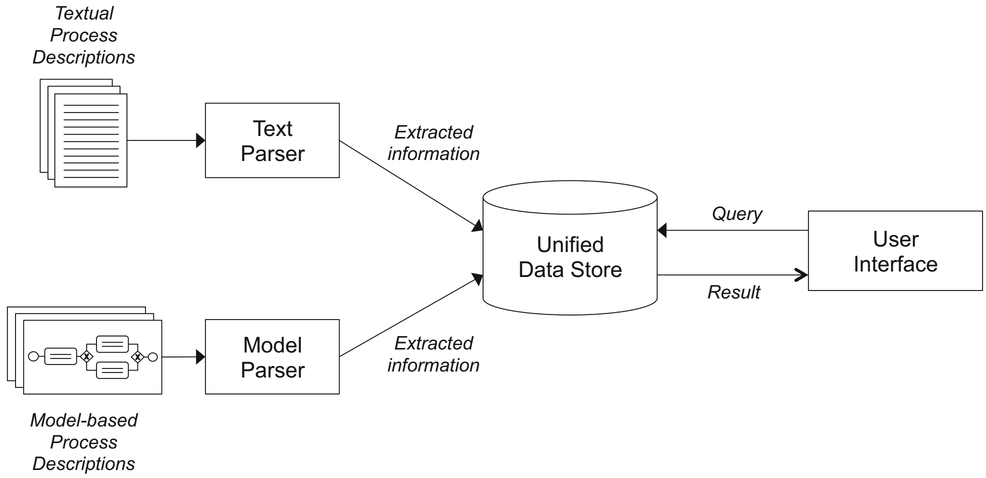


Fig. 2 Overview of proposed solution

3.1 Overview

To enable comprehensive process search, activity-related and behavioral information from both textual and model-based process descriptions must be stored in a unified way. Hence, two parsing components first extract the relevant information from the two input sources and then store it in the unified data store. Once the data store has been populated with all available process descriptions, it can be used for process search. To this end, a user interface provides the possibility to specify queries in a user-friendly manner. Figure 2 illustrates our architecture graphically.

In the subsequent sections, we describe this architecture in detail. Because of the predominant role of the unified data store, we begin with the specification of the unified format.

3.2 A unified format for integrating textual and model-based process descriptions

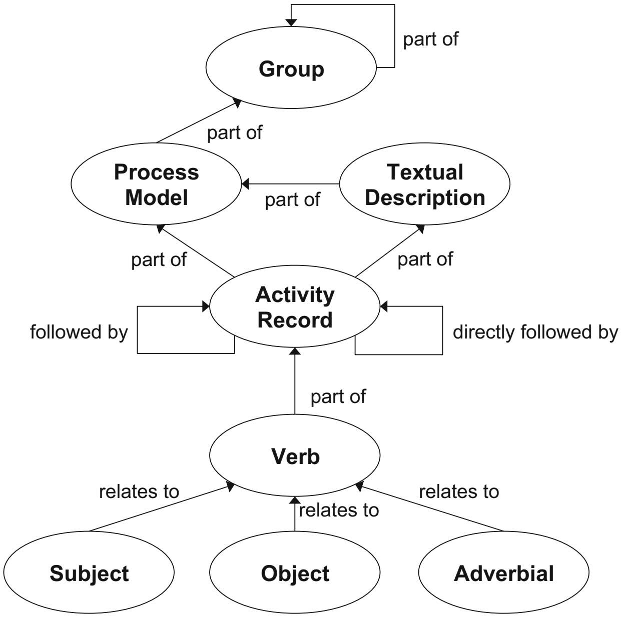


Fig. 3 Overview of unified data format

In this section, we define a unified data format for textual and model-based process descriptions, which is graphically illustrated in Fig. 3. When searching process-related artifacts, the search typically relates to the *activities* of the processes and the *interrelations* between these activities (i.e., their order). Therefore, the core of our unified format is a so-called *activity record*, which refers to a particular task or a step in a business process. The figure illustrates that an activity record might be part of a *process model* or a *textual description*. As indicated by the relations between the activity record, the process model, and the textual description, each process model and each textual description may contain several activity records. The interrelations between these activity records are captured by the *followed by* and *directly followed by* self-relations. Moreover, a process model may consist of several textual descriptions. As process models in industry are typically organized in hierarchical process architectures [36], our format allows to organize process models in *groups* and subgroups. In the remainder of this section, we discuss the notion of activity records, their interrelations, and the implementation of the data format in detail.

Activity records Figure 3 shows that activity records may consist of one or more *verbs*. Each verb relates to three optional entities: a *subject*, an *object*, and an *adverbial*. Together with verbs, these entities describe the four *semantic components* of an activity record. We define activity records in this decomposed manner because this format is compatible with activities that are described in process models as well as in textual process descriptions. Even though textual and model-based process descriptions vary considerably in the way they convey such semantics, the activities that they describe consist of the same semantic components.

Process models such as BPMN models consist of graphical representations of modeling constructs such as activities, events, and gateways [14]. Because *activities* are modeled using separate construct, individual activity records can be straightforwardly identified from a model. However, as pointed out by [29], the essential semantics of process activities are defined by their natural language labels. What is important is that these labels do not necessarily represent proper sentences [31]. As examples, consider the activity labels "*Opening of bank account for customer*" or "*Customer credit history evaluation*" from the BPMN model in Fig. 1. A study conducted by [38] revealed that, even though activity labels often do not represent grammatically correct sentences, they refer to three semantic components: an *action*, a *business object* on which the action is performed, and an optional *additional information fragment* that provides further details. As an example, consider the activity label "*Opening of new bank account for customer*." Although this label does not contain a verb, the noun "*opening*" implies the execution of the action "*to open*." Hence, "*opening*" would be classified as an action. Respectively, "*new bank account*" represents the business object, and "*for customer*" the additional information fragment. In the unified data format, we capture actions using a *verb*, the business object as the associated *object*, and the *adverbial* captures additional information fragments. Lastly, we store the actor, i.e., the resource, that performs the activity using the *subject* relation. In most process model notations, the actor is specified outside of the activity label. For example, in BPMN, *pool* and *lane* constructs are used to specify which actors perform which activities.

Unlike process models, textual process descriptions do not have an explicit notion of activities. Rather, activity records must be identified as part of the natural language sentences in a textual description. The descriptions of activity records occur as part of these paragraphs. For instance, the sentence "*Hence, the clerk collects the required documents and send the form to the customer per mail*" describes two actions performed by a clerk: *collecting documents* and *sending a form*. Nevertheless, in such sentences we can again identify the different components of an activity record. Consider the sentence "*The clerk opens a new bank account for the customer*." A grammatical analysis would reveal that this sentence contains the predicate "*opens*", the object "*bank account*," the subject "*clerk*," and the adverbial "*for the customer*." This example illustrates that the predicate corresponds to the action, the object to the business object, and the adverbial to the additional information fragment. Finally, a subject refers to the role executing the activity.

Activity interrelations The order in which process activities are executed represents a major element of a process-oriented perspective. To allow for comprehensive process search, it is important that the ordering relations between activities can be taken into account. As an example, consider a situation in which we want to retrieve all processes where a document must be manually signed before a purchase order can be send. In this case, it does not suffice to retrieve the processes in which "*sign document*" and "*send purchase order*" co-occur. It is also important that the former activity happens *before* the latter.

As depicted in Fig. 3, we capture two different activity interrelations: *followed by* and *directly followed by*. The former corresponds to the *weak order* relation $\succ$ [53]. Given two activity records x and y, a weak order relation $x \succ y$ indicates that during the execution of a process instance, x *can* occur before y. The directly followed by relations correspond to the so-called *alpha relations* $>$ [50]. For this relation type, $x > y$ denotes that an activity record x can occur directly before y. These two relation types can be used for different querying purposes. Furthermore, by storing these basic relations, it is also possible to store and query more complex interrelations.

To illustrate this, we consider the different behavioral relations that can be derived from the weak order relation: strict order, interleaving order, and exclusiveness. These relations are also referred to as *behavioral profile relations* [53]. The *strict order* relation $x \rightsquigarrow y$ is used to express that activity x cannot be executed after the execution of activity y. The *interleaving order* relation $x \mid\mid y$ states that x and y can be executed in an arbitrary order. Finally, the *exclusiveness* relation $x + y$ denotes that either activity x or activity y can be executed in a single process instance. These three relations can be derived by determining the existence of $x \succ y$ and $y \succ x$ for two given activity records. For example, the strict order $x \rightsquigarrow y$ follows from $x \succ y$ and $y \not\succ x$. An interleaving order $x \mid\mid y$ follows from $x \succ y$ and $y \succ x$, whereas exclusiveness results from $x \not\succ y$ and $y \not\succ x$ [53]. In a similar manner, it is possible to derive specific relations from the directly follows relation $>$.

Implementation Practically, we implement this unified data format by building on the Resource Description Framework (RDF), an XML-based specification developed by the World Wide Web Consortium (W3C)[1]. RDF describes data in the form of triples that consist of two entities and a relation between them. As an example, consider the relation *relates to* between an *object* and a *verb* in our unified data format. A possible RDF triple for this relation would be "*customer*," "*relates to*," "*reject*"[2]. Similarly, all other relations from the unified data format can be represented as RDF triples. The advantage of storing data in the RDF format is that it can

[1] http://www.w3.org/RDF/

[2] Note that we use a simplified syntax for RDF triples in order to provide readable and understandable illustrations.

be easily and effectively accessed and queried [8]. Hence, it greatly contributes to our goal of providing a technique for integrated search.

In the subsequent sections, we describe how textual process descriptions as well as process models can be automatically transformed into this unified data format.

3.3 Parsing textual process descriptions

The text parser component takes a textual description as input and automatically extracts the process information required for the unified data format. The procedure to extract and store activity records from textual descriptions consists of four subsequent steps:

1. *Extraction of sentences from textual descriptions*: Natural language descriptions are typically stored as PDF or Word files. The automated processing of these file formats is supported by several freely available tools such as Apache PDFBox[3] or Apache POI[4] and, hence, does not represent an obstacle. The challenge for extracting sentences from these documents is rather caused by recognizing where an individual sentence starts and where it ends, since a period is not only used at the end of a sentence but also in the context of abbreviations (e.g., "*M.Sc.*") [20]. We use the Stanford Parser [26], a highly accurate tool for natural language analysis, to automatically transform a text into a list of individual sentences.
2. *Linguistic analysis of sentences*: In the context of the linguistic analysis, we identify the grammatical entities such as subject, object, predicate, and adverbial for each sentence [35]. We also determine the relations between these entities, e.g., which verb relates to which object [12]. To illustrate the required steps and the associated challenges, consider the sentence "*Hence, the clerk collects the required documents and sends the form to the customer per mail*" from Fig. 1. It contains one subject ("*clerk*"), two predicates ("*collects*" and "*sends*"), two objects ("*documents*" and "*form*"), and two adverbials ("*to the customer*" and "*per mail*"). Furthermore, it is important to note the relations between these entities. The predicate "*collects*" relates to the object "*documents*," and the predicate "*sends*" relates to the object "*form*." To automatically determine these grammatical entities and their relations, we again make use of the Stanford Parser. Besides the recognition of sentence borders and the identification of parts of speech, the Stanford Parser is also capable of producing word dependencies [13]. As an example, consider the following two object-related dependencies that the Stanford Parser generates for the considered example sentence:

   dobj(collects-5, documents-8)
   dobj(sends-10, form-12)

   These so-called direct-object dependencies (dobj) specify which words the Stanford Parser considers to be objects and which predicates relate to them. Thus, the first dependency tells us that "*documents*" (position 8 in the sentence) is an object that relates to the predicate "*collects*" (position 5 in the sentence). Analogously, "*form*" is an object that relates to the predicate "*sends*." To make use of these generated dependencies, we developed an algorithm that automatically analyzes the Stanford Parser output and extracts the grammatical entities as well as their relations. Our component builds on the knowledge about existing dependencies (cf. [13]) and the consistent structure of these dependencies (name of the dependency followed by brackets that include two entities and their position). As a result, we are able to automatically obtain a set of grammatical entities and their relations from any given natural language sentence.
3. *Normalization of sentence components*: The words in sentences often do not occur in their base forms, i.e., verbs are not only used as infinitives and nouns are not always provided as singular nouns [44]. This becomes a problem when entities are compared in the context of a search operation [41]. For instance, "*send*," "*sends*," and "*sent*" all refer to the same verb. However, an automated string comparison would indicate that these words differ from each other. To deal with such cases, we use the lexical database WordNet [40] to convert all words into their base form, i.e., predicates into infinitive verbs and subject as well as objects into singular nouns. As a result, the predicate "*sends*" is transformed into "*send*."
4. *Transformation of sentence components into RDF*: Once the entities have successfully been extracted and transformed into their base forms, the information is stored as RDF [10]. To demonstrate this step, again consider the sentence "*Hence, the clerk collects the required documents and sends the form to the customer per mail*" from Fig. 1. For each predicate of the sentence, we create a *verb - activity record* RDF triple in order to capture the relation between the predicates and the sentence. The sentence is then represented by an activity record. Suppose this activity record has the identification number 2, then the respective RDF triples look as follows:

   ("*collect*", "*part of*", "*ActivityRecord2*")
   ("*send*", "*part of*", "*ActivityRecord2*")

   In addition, we need to link the subjects, objects, and adverbials to the respective verbs:

[3] https://pdfbox.apache.org/

[4] https://poi.apache.org/

("*clerk*", "*relates to*", "*collect*")
("*clerk*", "*relates to*", "*send*")
("*document*", "*relates*", "*collect*")
("*form*", "*relates*", "*send*")
("*to customer*", "*relates to*", "*send*")
("*per mail*", "*relates to*", "*send*")

To extract the interrelations between activity records, we use three consecutive steps:

1. *Pair-wise relation identification:* The extraction of activity interrelations from textual descriptions requires tailored natural language processing techniques [39]. For this purpose, we can build on existing *text-to-process model* generation approaches, such as [16,18,47]. These approaches employ heuristic-based techniques that recognize typical patterns used to express activity interrelations. Such patterns can be used to identify sequences, choices, and parallel executions. For instance, using such patterns it is possible to identify that the sentence "*Once the response from the customer is received, the clerk enters the data from the form into the system*" describes two subsequent process steps, i.e., "*receive response*" followed by "*enter data*." For brevity, we here omit further technical details of the relation extraction approach. The interested reader may consult the work by Friedrich et al. [16] for a detailed description of a state-of-the-art parsing technique.
2. *Propagation of relations:* The previous step yields interrelation between pairs of activity records that are described consecutively in the text. However, to allow users to also search for indirect relations, we have to propagate the *followed by* relations extracted in the previous step. For example, if we extract that $x \rightsquigarrow y$ and $y \rightsquigarrow z$, then we can combine these interrelations into $x \rightsquigarrow z$. We can perform this propagation due to the transitive properties of the strict order and interleaving order relations. After propagation, we have obtained a complete view on the interrelations specified in a textual description.
3. *Transformation of behavioral relations into RDF:* Once the activity interrelations have been extracted, they can be stored in RDF. As described in Sect. 3.2, we store behavioral relations using two basic *followed by* and *directly followed by* relations. Therefore, we transform the derivable relations, such as the *strict order* relation, using this basic format. Given a relation *ActivityRecord1* || *ActivityRecord2*, we need to store two relations in RDF:

   ("*ActivityRecord1*", "*followed by*", "*ActivityRecord2*")
   ("*ActivityRecord2*", "*followed by*", "*ActivityRecord1*")

In the next section, we describe how we extract the RDF triples from process models.

3.4 Parsing model-based process descriptions

This component expects a set of process models as input and automatically extracts the information required for the unified data format. Similar to the parsing of textual process descriptions, it consists of four subsequent steps:

1. *Extraction of activity labels from process models*: As opposed to textual process descriptions, which are typically available as PDF or Word files, process models in industry are created using several notations and different modeling tools [52]. As a result, process models are available in various file formats such as JSON, AML, EMPL. To deal with this variety, we build on the import functionality of the process platform proposed by [15]. This platform can import a wide range of different process model formats and store them in a suitable way. Once the process models were imported, the activity labels are extracted as a simple string with a single line of Java code.
2. *Linguistic analysis of activity labels*: Taking the extracted activity label strings as input, the linguistic analysis of activity labels aims at properly deriving the activity components such as action, business object, and additional information fragment [29]. As discussed in Sect. 3.2, the challenge is to automatically detect the varying grammatical structures, even if the activity label does not contain a proper verb. As an example, consider the activity "*Customer credit history evaluation*" from 1. For this label, it is necessary to automatically recognize that "*evaluation*" represents the action and "*customer credit history*" the business object. Even more challenging are ambiguous cases, in which more than a single interpretation of the label exists. As an ambiguous example, consider the activity label "*Plan Data Transfer*." This activity could instruct to "*plan*" a "*data transfer*" as well as to "*transfer*" a set of "*plan data*." To resolve such ambiguous cases and to properly derive the components from activity labels, we employ the label analysis technique introduced by [33]. This technique uses knowledge about possible linguistic structures and an analysis of the model context to decompose any activity label into its components. It takes an activity label as input and, respectively, returns the comprised action(s), business object(s), and additional information fragment(s).
3. *Normalization of activity label components*: Similar to proper natural language sentences, activities often contain inflected words, i.e., verbs occurring in the third person form or nouns used in the plural form. What is more, actions may even represent nouns (e.g., "*evaluation*" in "*Customer credit history evaluation*"). As pointed out for sentences, this has notable implications if two components are compared in the context of a search

operation. Hence, we apply the lexical database WordNet [40] also in activity labels to convert all actions into infinitive verbs and all nouns into singular nouns. As a result, the action of the activity label "*Customer credit history evaluation*" is transformed into "*evaluate*."

4. *Transformation of activity label components into RDF*: The storage of the extracted and normalized components as RDF works analogously to the storage of the sentence entities. We demonstrate this step using the activity "*Opening of new bank account for customer*." As a result of applying the previous steps, we identified the action "*open*," the object "*bank account*," and the addition "*for customer*." Suppose the resulting activity record has the identification number 3, then the RDF triples look as follows:

   ("*open*", "*part of*", "*ActivityRecord3*")
   ("*bank account*", "*relates to*", "*open*")
   ("*for customer*", "*relates to*", "*open*")

   The example triples illustrate that the action is linked to the activity by using the *verb - activity record* triple. The business object and the addition are, respectively, associated with the verb by using the *object–verb* and the *adverbial–verb* relation.

To extract the relations between activity records from a process model, we use two consecutive steps:

1. *Pair-wise relation identification:* To extract behavioral relations from process models, we employ existing techniques. The directly follows relation $>$ can be directly extracted from the process model. This relation includes any activity pair that is directly connected, possibly through a gateway, with each other. In order to extract the weak order relation $\succ$, we use the method described in [53].
2. *Transformation of behavioral relations into RDF:* Once the directly follows and weak order relations have been extracted from the process, we can directly enter them into the database. The resulting relations have the same format as the relations that are extracted from textual process descriptions. For example, the following example shows the insertion of two activity records, which defines that *ActivityRecord1* can be directly followed by *ActivityRecord2*. Because we do not insert the inverse relation, it becomes evident that these two activity records cannot directly follow each other in reverse order.

   ("*ActivityRecord1*", "*directly followed by*", "*Activity Record2*")
   ("*ActivityRecord1*", "*followed by*", "*ActivityRecord2*")

In the next section, we show how we query the extracted data.

```
1  PREFIX ps: <http://www.processsearch
       .com/Property/>
2  SELECT ?processName
3  WHERE
4  {
5       ?verb ps:Type "Verb" .
6       ?verb ps:Label "send" .
7
8       ?object ps:RelatesTo ?verb .
9       ?object ps:Type "Object" .
10      ?object ps:Label "form" .
11
12      ?verb ps:PartOf ?activityRecord
          .
13      ?activityRecord ps:PartOf ?
            processModel .
14      ?processModel ps:Label ?
            processName .
15 }
```

Listing 1 Exemplary SPARQL query to retrieve data from the RDF database

3.5 Querying the unified data store

In order to query the extracted RDF triples, we use SPARQL (Simple Protocol and RDF Query Language). In essence, SPARQL is similar to SQL but is specifically designed to query RDF data. As an example, consider the following SPARQL query, which retrieves all process models and textual process descriptions that contain an activity record relating to the verb "*send*" and the object "*form*":

The example from Listing 1 shows that a SPARQL query has the basic structure of an SQL query, i.e., it follows the *select-from-where* pattern. Before the actual query, however, it is required to define where the data model definition can be found (line 1). As SPARQL was designed for the semantic web, this is done via a Unified Resource Identifier (URI). In this example, we use the URI http://www.processsearch.com/Property/ for illustration purposes. After the definition of this prefix, the actual query starts. Line 2 specifies that we are interested in all process names of process models that fulfill the requirements stated as RDF triples in the block below. Using the variable *?verb*, we define that there must be a *Verb* according to our data model that carries the label "*send*" (lines 5 and 6). Moreover, we use the variable *?object* to define that *?verb* must be related to an *Object* that carries the label "*form*" (lines 8-10). Finally, we define that we are only interested in process models that contain an activity record that relates to an entity Verb as specified in *?verb*.

This exemplary query illustrates that an RDF-based unified data store can be easily queried for information we are interested in. To provide the users of our technique with an intuitive possibility to search, we implemented a graphical interface in which users can specify the verbs, objects, subjects, and adverbials of the process descriptions they would like to search for. The input from the graphical user interface is then automatically inserted into a SPARQL query as provided above. As a result, the user can perform any search based on these four entities and does not have to deal with any technical details.

4 Evaluation

In this section, we demonstrate the applicability of our integrated search technique. To this end, we implemented it as a Java prototype and applied it to the process repository of a large Austrian bank. Our goal is to demonstrate the added value that is associated with searching textual descriptions on top of process models. Section 4.1 discusses the setup of our evaluation experiment. Section 4.2 introduces the data, i.e., the process model collection we use. Section 4.3 explains the prototypical implementation of our technique. Section 4.4 presents the results. Section 4.5 assesses the quality of the results. Section 4.6 discusses implications and limitations of our technique.

4.1 Setup

To identify search scenarios that can show the applicability and impact of our technique in practice, we collaborated with a large Austrian bank. Based on a series of group discussions with the Business Process Management department of this bank, we identified three business-relevant search scenarios which are of particular interest:

1. *Search for media disruptions*: Media disruptions occur when the information-carrying medium is changed, for example, when a clerk enters the data from a letter into an information system. Media disruptions should be avoided when possible because they are a major source of errors. These errors, among others, follow from proneness to (manual) conversion mistakes. Hence, our evaluation partner had a considerable interest in identifying media disruptions in their process landscape. To design a suitable query, we built on the insights from [5]. In a study on weakness patterns, they found that media disruptions are indicated by the actions "*print*" and "*scan*" as well as the activity "*Enter data*."
2. *Search for manual activities*: Manual activities are inevitable in most business processes. However, as automation is often associated with saving costs, the identification of automation candidates represents a key task in business process improvement [34]. Thus, our evaluation partner was also interested in identifying which automation potential their process repository exhibits. In their study, [5] also identified patterns for manual activities. They found that manual activities are indicated by the actions "*document*," "*record*," and "*calculate*" as well by combinations of the actions "*verify*" and "*archive*" with the business objects "*document*" and "*information*."
3. *Check of customer signature*: To prevent fraud, it is of paramount importance to verify the signature of the customer. While this essentially represents a standard procedure in the daily business of a bank, it is also important that this procedure is consistently reflected in the process documentation. To check for instances where the customer signature is checked (or not checked), we make use of the behavioral relations. We search for a sequence of activity records where the first activity record refers to the subject "*customer*" and the verb "*sign*," and the second activity record refers to the verb "*check*" and the object "*signature*." By comparing the results of this search against the results of simple search for a combination of the subject "*customer*" and the verb "*sign*," we can conclude about the number of cases that do not highlight need for checking the customer signature.

We use queries based on the patterns discussed above to demonstrate the capabilities of our approach and to show the importance of taking text-based process descriptions into consideration.

4.2 Data

The process repository of the Austrian bank we use for applying the above-defined queries consists of 1,667 process models created using the Event-driven Process Chain (EPC) notation. The process models cover various aspects of the banking business including the opening of accounts, the management and selling of financial products, as well as customer relationship management. On average, the process models contain 6.5 activities per model. The smallest model contains 1, and the largest contains 181 activities. In addition to the process models, the repository contains 119 textual process descriptions in the PDF format. The textual descriptions complement the process models and mainly concern the area of credit management. Due to existing overlaps between the process models in the repository, a single textual description can be referred to by multiple process models. The size of these complementary process descriptions ranges from 119 to 60,558 words. Most of the description is rather long, resulting in an average size of 13,130 words. The language of both the process model elements and textual process descriptions is German. Table 1 summarizes the key characteristics of our data collection.

Table 1 Characteristics of process repository

No. of process models: 1667	
No. of activities (minimum)	1
No. of activities (maximum)	181
No. of activities (median)	5
No. of activities (average)	6.5
No. of activities (standard deviation)	9.38
No. of textual process descriptions: 119	
No. of words (minimum)	119
No. of words (maximum)	60,558
No. of words (median)	10,688
No. of words (average)	13,130
No. of words (standard deviation)	11,743

4.3 Implementation

We implemented the approach defined in Sect. 3 as a Java prototype. To be able to deal with German process models and process descriptions, we integrated the German package of the Stanford Parser [26], the German component of the label analysis technique from [33], and a German implementation of WordNet called GermaNet [22]. In addition to these techniques, we use the Apache PDFBox to process PDF files and the import functionality from [15] to process different process model formats. Finally, to store the extracted RDF triples, we use the Apache Jena component TDB[5], which is a database optimized for RDF storage and querying. For running the experiments, we deployed the techniques on a MacBook Air with a 1.4 GHz Intel Core i5 processor and 8GB RAM, running on Mac OS 10.10.2 and Java Virtual Machine 1.8.

4.4 Results

The results for the search for media disruptions and the search for manual activities are illustrated in Figs. 4 and 5. The figures show the aggregated total for the two search scenarios and the disaggregated number of retrieved processes (i.e., the number of descriptions referring to a specified business process) for each weakness pattern (e.g., Verb = "*print*" or Verb = "*document*"). The light gray bars indicate the number of processes we retrieved from searching the model-based process descriptions, and the dark gray bars indicate the number of processes we retrieved from searching the model-based as well as the textual process descriptions. The results illustrate that the number of retrieved processes considerably increases when querying both process description types. Interestingly, that does not only hold for the total of both search scenarios, but also for each of the weakness patterns as, for instance, for the verb "*print*" in the media disruption scenario or for the verb "*document*" in the manual activity search scenario. Altogether, the number of processes that are retrieved from the process repository increases from 83 to 151 for the media disruption scenario (an increase of 81.9%) and from 213 to 359 (an increase of 68.5%) for the manual activity scenario[6].

Figure 6 shows the results for the third search scenario, the check of the customer's signature. The results again illustrate that the additional consideration of the textual process documentation yields a higher number of retrieved processes (109 versus 76). What is more, it illustrates the value of being able to incorporate behavioral aspects. The number of processes where a customer signs a document is higher than the number of processes where this step is followed by a check of the signature (109 versus 81). The results of this search, therefore, reveal that 28 process documentations (out of which 14 represent textual process documentations) should be checked for consistency.

A detailed analysis of the results revealed that there is no overlap between the processes retrieved from the model-based and the textual descriptions in any of the search scenarios. This shows that the details of some processes are fully described by process models, while the details of others are only captured by textual descriptions. A look into the process models linking to these detailed textual descriptions showed that they typically contain very high-level activities. This does not only explain the lack of overlap between the result sets but also again highlight the importance of considering both types of process descriptions.

4.5 Assessment of result quality

In the previous section, we showed that the consideration of textual process descriptions results in a higher number of identified processes. While this highlights the advantage of searching textual and model-based process descriptions using our technique, it does not provide a full picture of the quality of our search results. Traditionally, the quality of search techniques in information retrieval is assessed using the metrics precision, recall, and F-measure [3]. In our context, precision is the number of relevant and retrieved processes divided by the total number of retrieved processes. Recall is the number of relevant and retrieved processes divided by the total number of relevant processes. The F-measure is the harmonic mean between the two.

[5] https://jena.apache.org/

[6] Note that because a single textual description can be referred to by several process models, the identification of one relevant textual document may yield multiple relevant process models. This explains why the increase in the number of retrieved processes might be even higher than the total number of textual process descriptions.

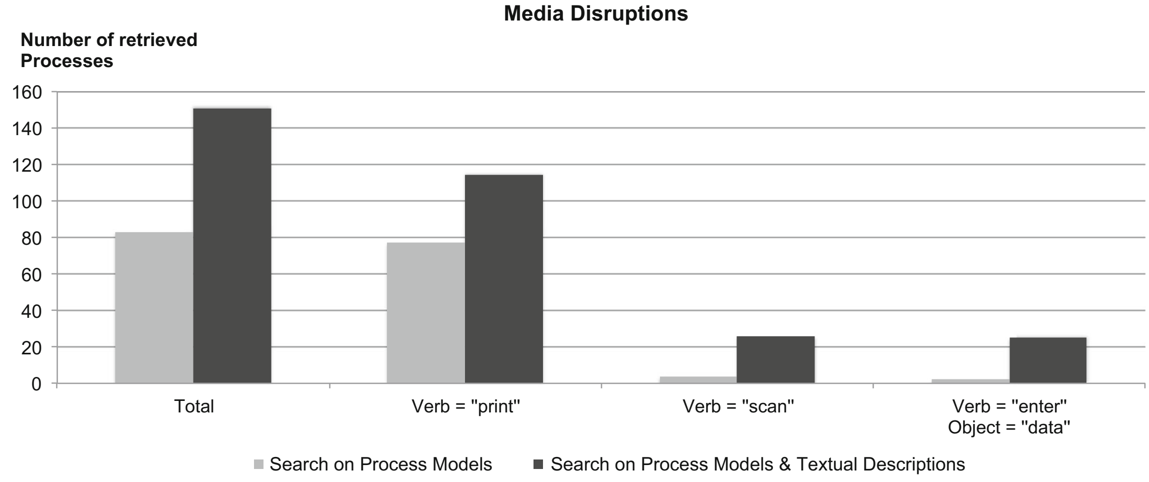


Fig. 4 Results for media disruptions search

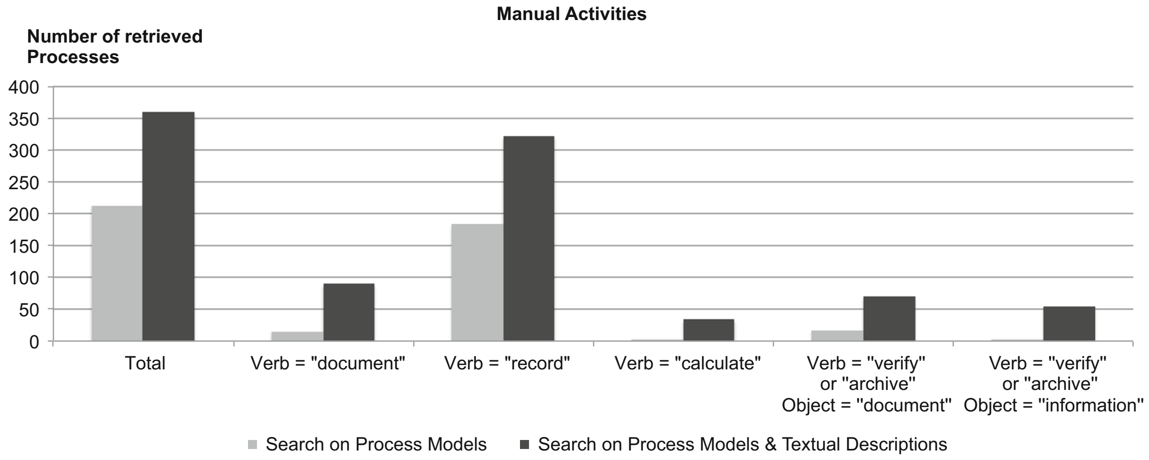


Fig. 5 Results for manual activities search

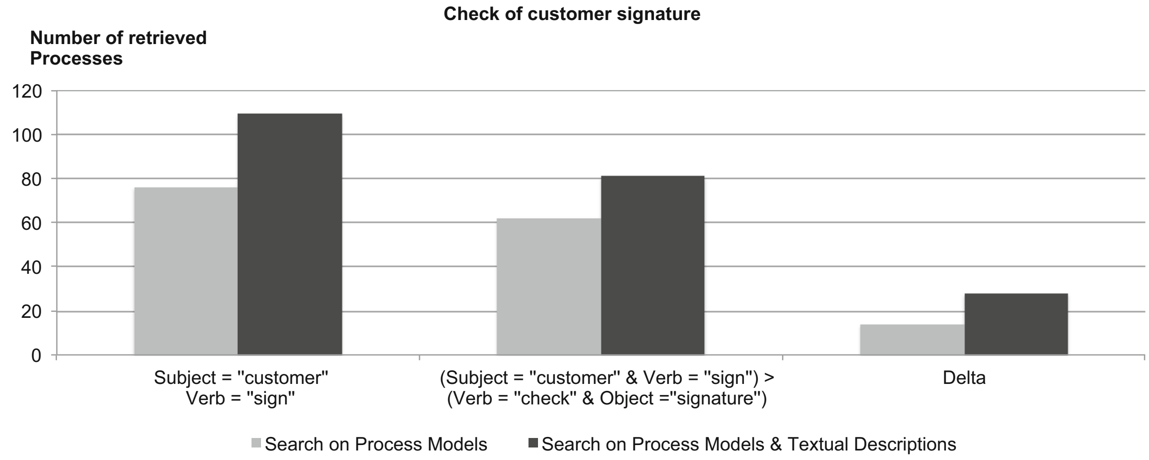


Fig. 6 Results for check of customer's signature search

A central challenge when using precision and recall is how to define the notion of *relevance*. It is important to recognize that relevance should be assessed relative to an *information need*, not to a query [9]. This means that, for instance, we should not assess whether a query for the verb "*scan*" indeed returns processes that contain "*scan*," but rather whether the

Table 2 Precision of results for the three search scenarios

Query	Precision	
	PM	PM & TD
V=“*print*”	0.99	0.98
V=“*scan*”	0.98	0.96
V=“*enter*” & O=“*data*”	1.00	1.00
V=“*document*”	0.99	0.98
V=“*record*”	0.99	0.98
V=“*calculate*”	1.00	1.00
V=“*verify*” / “*archive*” & O=“*document*”	1.00	1.00
V=“*verify*” / “*archive*” & O=“*information*”	1.00	1.00
S=“*customer*” & V=“*sign*”	1.00	0.99
S=“*customer*” V=“*sign*” > V=”check” & O=“*signature*”	1.00	1.00

process indeed contains a media disruption where *scanning* is used to transform a physical into a digital document. In terms of the precision of our approach, issues in this respect can occur in the form of false positives caused by, among others, *polysemous* words, i.e., words with several meanings. The verb “*scan*,” for instance, can refer to the activity of systematically examining something (i.e., reading a document) or the activity of producing a digital copy of a document. Results related to the former meaning are not relevant to our query, whereas results related to the latter meaning are. Table 2 shows the precision for the three search scenarios.

The precision values from Table 2 show that the precision is generally very high. However, we encountered false positive cases for the query terms “*print*,” “*scan*,” “*document*,” “*record*,” and “*sign*.” Most of them related to instances of polysemy. As an example, consider the activity “*Scan the database for relevant customers*,” which clearly does not relate to a meaning of “*scan*” that can be associated with a media disruption. More detailed textual process description also contains instances of the query terms that do not relate to an actual activity. As an example, consider the sentence “*When a message box stating ‘Printing not possible’ appears, ignore it and simply continue*.” For such cases, our technique would mark the process as relevant although it does not actually contain an activity related to “*printing*.”

Analyzing the precision results in more detail provides three specific insights. First, precision is mainly affected by queries of potentially polysemous verbs (note, for instance, that “*calculate*” can hardly have a wrong meaning when queried). Second, precision slightly decreases when including the textual process descriptions in the search. This can be explained by the increasing level of detail and the likelihood of encountering a query term that does not relate to an actual activity. Third, queries combining verbs and objects often have a perfect precision. This is caused by the disambiguating function of the object. It is simply less likely that the same verb has two completely different meanings for the same object (e.g., a “document”). This means that false positives can be effectively reduced by formulating more specific queries, such as queries that include objects.

Given the size of the process repository and the included textual process descriptions, it is infeasible to manually assess the relevance of all processes in the collection to our search scenarios. Therefore, given the lack of an exact number of relevant results, we cannot provide a full quantification of the *recall* of our approach for the considered search scenario. However, based on the number of returned results and the precision values, we can clearly observe that the recall of the queries that include textual descriptions, i.e., *PM & TD*, is considerably higher than the queries that exclude these from considerations. For example, for the *media disruptions* use case, *PM & TD* returns 151 processes, whereas just *PM* returns 83 processes. From the assessment of the precision of these results, we learned that these, respectively, include 147 and 81 relevant processes. Therefore, even though we cannot compute the exact recall value, we know that the recall of *PM & TD* is 1.81 times (i.e., 147 divided by 81) greater than the recall for *PM*. Similar patterns can be observed for the recall of the other two search scenarios.

All in all, these insights highlight the value of including textual process descriptions in the search.

4.6 Discussion

The evaluation of our technique with three business-relevant search scenarios and a process repository from the large Austrian bank illustrated that the accuracy of our search results is considerably higher when textual process descriptions are taken into account. Hence, this evaluation did not only demonstrate the applicability of our technique but also its practical relevance.

The practical relevance of our technique is further demonstrated by the way it was perceived by our evaluation partner. The business analysts from the bank considered the results obtained by our technique so useful for their analyses that they decided to integrate our technique with their ARIS platform. To achieve this, they set up a script that updates the database associated with our technique on a daily basis. In this way, search operations can be conducted in a considerably more efficient way than before.

Against this background, our work has the following implications. From a research perspective, this work provides the means for storing textual and model-based process information in a unified way. It also provides the techniques to extract the required information from the two process description types. Among others, these contributions can enable existing search and analysis techniques to additionally consider textual process descriptions. From a practice perspective, our techniques enable organizations to conduct more comprehensive search operations. The value of automated search and analysis techniques is increased because integrated techniques enable the identification of relevant processes that are not retrieved by techniques limited to model-based descriptions.

While the results presented in this paper are promising, they also have to be discussed in the light of some limitations. First, we would like to stress that the investigated process repository is not statistically representative. For instance, process repositories from other companies may consist of more or also fewer textual descriptions than the one we investigated. However, what we would like to point out is that the technique presented in this paper does not rely on any specifics we encountered in the repository of our evaluation partner as, for instance, the EPC notation. Instead, we provide a generic technique that enables organizations to search textual and model-based descriptions in an integrated way. While the individual value that is associated with the possibility may differ, we are convinced that it always represents an advantage to be able to take textual process descriptions into account. Second, it should be noted that our technique cannot guarantee that all relevant information is identified. On the one hand, the user has to define proper key words. On the other hand, our technique can only search the information that is documented explicitly in one of the addressed description types and actually available to the search engine. This means that the presented technique can only identify all relevant pieces of information if the respective documents are linked to the database. Collecting and linking relevant documents, therefore, is a necessary preparation step. Third, the current data format uses the adverbial component to capture all information that is provided on top of the verb, subject, and object. There might be use cases where a more fine-granular consideration is conducive. Since this, however, would result in a more complex data structure, we decided to stick to the more manageable solution presented in this paper. Fourth, our technique is currently implemented in German and English only. While the employed natural language processing tools are available for many languages (see, e.g., [37]), the adaptation is associated with a certain implementation effort. However, in the light of the usefulness of our approach, we are convinced that this effort can be well justified if other languages need to be covered.

5 Conclusion

In this paper, we introduced a comprehensive search technique that allows the user to find information in textual as well as in model-based process descriptions. The technique builds on the transformation of textual and model-based process descriptions into a unified data format that integrates the activity-related and behavioral information from both sources. We implemented the technique as a Java prototype that stores the extracted data in an RDF database and provides the user with a graphical interface to specify queries. An evaluation with a large Bank from Austria showed that our solution can be successfully applied in industry and that the additional consideration of textual process descriptions indeed increases the number of identified processes.

From a research perspective, the provided technique provides the foundations for integrating textual and model-based information. To the best of our knowledge, we are the first to define in integrated data format that allows to combine the textual information these two artifact types. Hence, our technique can inform existing process search techniques and may help to increase their scope. From a practical perspective, our technique helps organization to perform more comprehensive search operations. As demonstrated in the evaluation, textual sources may contain equally relevant information about processes as process models.

In future work, we plan to extend our approach with respect to the data perspective. This is particularly challenging for textual process descriptions since data-related aspects are often described in a rather implicit manner. Another valuable extension we have in mind is incorporating semantic technology. In this way, users can also find processes and activities that contain words that are synonymous to the search terms. Besides these extensions of our search technique, we plan to investigate whether the unified data format can also help to solve other research problems as, for instance, the checking of consistency between textual and model-based process descriptions.

References

1. Awad, A., Decker, G., Weske, M.: Efficient compliance checking using BPMN-Q and temporal logic. In: Dumas, M., Reichert, M., Shan, M.C. (eds.) Business Process Management. BPM 2008. Lecture Notes in Computer Science, vol 5240. Springer, Berlin, Heidelberg (2008)
2. Awad, A., Polyvyanyy, A., Weske, M.: Semantic querying of business process models. In: Enterprise Distributed Object Computing Conference, 2008. EDOC'08. 12th International IEEE, pp. 85–94. IEEE (2008)
3. Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press/Addison-Wesley, Boston (1999)
4. Baier, T., Mendling, J.: Bridging abstraction layers in process mining by automated matching of events and activities. In: Daniel, F., Wang, J., Weber, B. (eds.) Business Process Management. Lecture Notes in Computer Science, vol 8094. Springer, Berlin, Heidelberg (2013)
5. Becker, J., Bergener, P., Räckers, M., Weiß, B., Winkelmann, A.: Pattern-based semi-automatic analysis of weaknesses in semantic business process models in the banking sector. ECIS 2010 Proceedings 156 (2010)
6. Becker, J., Delfmann, P., Herwig, S., Lis, L., Stein, A.: Formalizing linguistic conventions for conceptual models. In: Conceptual Modeling—ER 2009, pp. 70–83. LNCS, Springer, Berlin, Heidelberg (2009)
7. Breuker, D., Pfeiffer, D., Becker, J.: Reducing the variations in intra- and interorganizational business process modeling - an empirical evaluation. In: Proceedings of the International Tagung Wirtschaftsinformatik (2009)
8. Candan, K.S., Liu, H., Suvarna, R.: Resource description framework: metadata and its applications. SIGKDD Explor. Newsl. **3**(1), 6–19 (2001)
9. Christopher, D.M., Prabhakar, R., Hinrich, S.: Introduction to information retrieval. Introd. Inf. Retr. **151**, 177 (2008)
10. World Wide Web Consortium: RDF 1.1 Concepts and Abstract Syntax. https://www.w3.org/TR/rdf11-concepts/
11. Davies, I., Green, P., Rosemann, M., Indulska, M., Gallo, S.: How do practitioners use conceptual modeling in practice? Data Knowl. Eng. **58**(3), 358–380 (2006)
12. De Marneffe, M.C., MacCartney, B., Manning, C.D., et al.: Generating typed dependency parses from phrase structure parses. In: Proceedings of LREC. vol. 6, pp. 449–454. Genoa Italy (2006)
13. De Marneffe, M.C., Manning, C.D.: The Stanford typed dependencies representation. In: Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation. pp. 1–8. Association for Computational Linguistics (2008)
14. Dumas, M., Rosa, M., Mendling, J., Reijers, H.: Fundamentals of Business Process Management. Springer, Berlin (2013)
15. Eid-Sabbagh, R.H., Kunze, M., Meyer, A., Weske, M.: A platform for research on process model collections. In: Mendling, J., Weidlich, M. (eds.) Business Process Model and Notation. Lecture Notes in Business Information Processing, vol. 125, pp. 8–22. Springer, Berlin, Heidelberg (2012)
16. Friedrich, F., Mendling, J., Puhlmann, F.: Process Model Generation from Natural Language Text. In: Proceedings of the 23rd International Conference on Advanced Information Systems Engineering, LNCS, vol. 6741, pp. 482–496. Springer (2011)
17. Friedrich, F.: Automated Generation of Business Process Models from Natural Language Input. Master's thesis, Humboldt Universität zu, Berlin (2010)
18. Ghose, A.K., Koliadis, G., Chueng, A.: Process discovery from model and text artefacts. In: Proceedings of the IEEE Congress on Services. pp. 167–174. IEEE Computer Society (2007)
19. Gonçalves, J., Santoro, F.M., Baiao, F.A.: Business process mining from group stories. In: Computer Supported Cooperative Work in Design, 2009. CSCWD 2009. 13th International Conference on. pp. 161–166. IEEE (2009)
20. Grefenstette, G.: Tokenization. In: van Halteren, H., (eds.) Syntactic Wordclass Tagging. Text, Speech and Language Technology, vol 9. Springer, Dordrecht (1999)
21. Gruhn, V., Laue, R.: Detecting common errors in event-driven process chains by label analysis. Enterp. Model. Inf. Syst. Archit. **6**(1), 3–15 (2011)
22. Hamp, B., Feldweg, H.: Germanet—a lexical-semantic net for German. In: Proceedings of ACL Workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications, pp. 9–15 (1997)
23. Hevner, A., March, S., Park, J., Ram, S.: Design science in information systems research. MIS Q. **28**(1), 75–105 (2004)
24. Indulska, M., Green P., Recker, J., Rosemann, M.: Business process modeling: perceived benefits. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., de Oliveira, J.P.M. (eds.) Conceptual Modeling - ER 2009. ER 2009. Lecture Notes in Computer Science, vol 5829. Springer, Berlin, Heidelberg (2009)
25. Jin, T., Wang, J., Wu, N., La Rosa, M., ter Hofstede, A.H.M.: Efficient and accurate retrieval of business process models through indexing. In: Meersman, R., Dillon, T., Herrero, P. (eds.) On the Move to Meaningful Internet Systems: OTM 2010. OTM 2010. Lecture Notes in Computer Science, vol 6426. Springer, Berlin, Heidelberg (2010)
26. Kunze, M., Weidlich, M., Weske, M.: Behavioral similarity - a proper metric. In: Rinderle-Ma, S., Toumani, F., Wolf, K., (eds.) Business Process Management. BPM 2011. Lecture Notes in Computer Science, vol 6896. Springer, Berlin, Heidelberg (2011)
27. Koschmider, A., Blanchard, E.: User assistance for business process model decomposition. In: Proceedings of the 1st IEEE International Conference on Research Challenges in Information Science. pp. 445–454 (2007)
28. Kunze, M., Weidlich, M., Weske, M.: Behavioral similarity—a proper metric. In: Business Process Management, pp. 166–181. Springer (2011)
29. Leopold, H.: Natural Language in Business Process Models: Theoretical Foundations, Techniques, and Applications, LNBIP, vol. 168. Springer, Berlin (2013)
30. Leopold, H., van der Aa, H., Pittke, F., Raffel, M., Mendling, J., Reijers, H.A.: Integrating textual and model-based process descriptions for comprehensive process search. In: BPMDS 2016, EMMSAD 2016: Enterprise, Business-Process and Information Systems Modeling. pp. 51–65. Springer International Publishing (2016)
31. Leopold, H., Eid-Sabbagh, R.H., Mendling, J., Azevedo, L.G., Baião, F.A.: Detection of naming convention violations in process models for different languages. Decis. Support Syst. **56**, 310–325 (2013)
32. Leopold, H., Mendling, J.: Automatic derivation of service candidates from business process model repositories. In: Abramowicz, W., Kriksciuniene, D., Sakalauskas, V. (eds.) Business Information Systems. BIS 2012. Lecture Notes in Business Information Processing, vol 117. Springer, Berlin, Heidelberg (2012)
33. Leopold, H., Smirnov, S., Mendling, J.: On the refactoring of activity labels in business process models. Inf. Syst. **37**(5), 443–459 (2012)
34. Limam Mansar, S., Reijers, H.A.: Best practices in business process redesign: use and impact. Bus. Process Manag. J. **13**(2), 193–213 (2007)
35. Loberger, G.J., Welsh, K., Shoup, K.: Webster's New World English Grammar Handbook. Wiley, Hoboken (2001)

36. Malinova, M., Leopold, H., Mendling, J.: An empirical investigation on the design of process architectures. Wirtschaftsinformatik Proceedings 75 (2013)
37. McDonald, R., Nivre, J., Quirmbach-Brundage, Y, Goldberg, Y., Das, D., Ganchev, K., Hall, K., Petrov, S., Zhang, H., Tackstrom, O., Bedini, C., Bertomeu Castelló, N., Lee, J.: Universal Dependency Annotation for Multilingual Parsing. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, vol 2. Association for Computational Linguistics, Sofia, Bulgaria, pp 92–97
38. Mendling, J., Reijers, H.A., Recker, J.: Activity labeling in process modeling: empirical insights and recommendations. Inf. Syst. **35**(4), 467–482 (2010)
39. Mendling, J., Leopold, H., Pittke, F.: 25 challenges of semantic process modeling. Int. J. Inf. Syst. Softw. Eng. Big Co. (IJISEBC) **1**(1), 78–94 (2015)
40. Miller, G., Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
41. Mladenic, D.: Learning word normalization using word suffix and context from unlabeled data. In: Proceedings of the Nineteenth International Conference on Machine Learning. pp. 427–434. Morgan Kaufmann Publishers Inc. (2002)
42. Peffers, K., Tuunanen, T., Rothenberger, M., Chatterjee, S.: A design science research methodology for information systems research. J. Manag. Inf. Syst. **24**(3), 45–77 (2007)
43. Pittke, F., Leopold, H., Mendling, J.: Automatic detection and resolution of lexical ambiguity in process models. IEEE Trans. Softw. Eng. **41**(6), 526–544 (2015)
44. Plag, I., Arndt-Lappe, S., Braun, M., Schramm, M.: Introduction to English Linguistics. Walter de Gruyter GmbH & Co KG, Berlin (2015)
45. Qiao, M., Akkiraju, R., Rembert, A.J.: Towards efficient business process clustering and retrieval: combining language modeling and structure matching. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) Business Process Management. BPM 2011. Lecture Notes in Computer Science, vol 6896. Springer, Berlin, Heidelberg (2011)
46. Simon, H.A.: The Sciences of the Artificial, 3rd edn. MIT Press, Cambridge (1996)
47. Sinha, A., Paradkar, A.: Use cases to process specifications in business process modeling notation. In: Proceedings of the IEEE International Conference on Web Services. pp. 473–480. IEEE (2010)
48. van der Aa, H., Leopold, H., Mannhardt, F., Reijers, H.A.: On the fragmentation of process information: challenges, solutions, and outlook. In: BPMDS'15 Working Conference (2015)
49. van der Aa, H., Leopold, H., Reijers, H.A.: Dealing with behavioral ambiguity in textual process descriptions. In: International Conference on Business Process Management. pp. 271–288. Springer (2016)
50. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9), 1128–1142 (2004)
51. van der Vos, B., Gulla, J.A., van de Riet, R.: Verification of conceptual models based on linguistic knowledge. Data Knowl. Eng. **21**(2), 147–163 (1997)
52. vom Brocke, J., Mendling, J.: Frameworks for business process management: a taxonomy for business process management cases. In: vom Brocke, J., Mendling, J. (eds.) Business Process Management Cases. Management for Professionals. Springer, Cham (2018)
53. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. IEEE Trans. Softw. Eng. **37**(3), 410–429 (2011)
54. Yan, Z., Dijkman, R., Grefen, P.: Fast business process similarity search. Distrib. Parallel Databases **30**(2), 105–144 (2012)

Dr. Henrik Leopold is an Assistant Professor at the Department of Computer Science at the Vrije Universiteit Amsterdam. His research interests include business process analysis using natural language processing and machine learning techniques, process model matching, and conformance checking. He has published over 50 research papers and articles, among others, in Data & Knowledge Engineering, Decision Support Systems, IEEE Transactions on Software Engineering, and Information Systems. His doctoral thesis received the German Targion Award 2014 for the best dissertation in the field of strategic information management.

Han van der Aa is a Ph.D. candidate in the Business Informatics group at the Vrije Universiteit Amsterdam, the Netherlands. He received his M.Sc. degree in Business Information Systems from the Eindhoven University of Technology in 2013. His research interests include business process modeling, process model matching, natural language processing, and compliance checking. His research has been published, among others, in Decisions Support Systems and Information Systems.

Fabian Pittke is a research assistant with the Institute for Information Business at Wirtschaftsuniversität Wien (WU Vienna), Austria. He received an MSc degree in business informatics from the Institute of Information Systems (IWI), Universität des Saarlandes, Germany, in 2010. He has been a research fellow with Humboldt-Universität zu Berlin until 2012 and is, since then, an external research fellow at the Institute for Information Business at Wirtschaftsuniversität Wien. His research focuses on linguistic aspects of process models. His research interests include business process modeling and natural language processing techniques.

Manuel Raffel is a Master student of Information Systems at the Wirtschaftsuniversität Wien (WU Vienna), Austria. There he also worked as a student assistant at the Institute for Information Business. His main responsibility was software development in the area of Business Process Management. Recently, he returned to his professional career as an IT Project Manager.

Jan Mendling is a Full Professor with the Institute for Information Business at Wirtschaftsuniversität Wien (WU Vienna), Austria. His research interests include various topics in the area of business process management and information systems. He has published more than 300 research papers and articles, among others in ACM Transactions on Software Engineering and Methodology, IEEE Transaction on Software Engineering, Information Systems, Data & Knowledge Engineering, and Decision Support Systems. He is member of the editorial board of seven international journals, member of the board of the Austrian Society for Process Management (http://prozesse.at), one of the founders of the Berlin BPM Community of Practice (http://www.bpmb.de), organizer of several academic events on process management, and member of the IEEE Task Force on Process Mining. His Ph.D. thesis has won the Heinz-Zemanek-Award of the Austrian Computer Society and the German Targion Award for dissertations in the area of strategic information management.

Hajo A. Reijers is a Full Professor at the Vrije Universiteit Amsterdam, where he heads the Information Management & Software Engineering group of the Computer Science department. He is also a part-time, Full Professor at the Department of Mathematics & Computer Science of Eindhoven University of Technology. His expertise is in enterprise systems, business process management, process mining, conceptual modeling, and workflow technology. Reijers has published over 150 scientific papers, chapters in edited books, and articles in professional journals.