Towards Automating the Detection of Event Sources

Nico Herzberg, Oleh Khovalko, Anne Baumgrass, and Mathias Weske

Hasso Plattner Institute at the University of Potsdam
Prof.-Dr.-Helmert-Straße 2-3, 14482 Potsdam
{nico.herzberg,oleh.khovalko,anne.baumgrass,
mathias.weske}@hpi.uni-potsdam.de

Abstract. During business process execution, various systems and services produce a variety of data, messages, and events that are valuable for gaining insights about business processes, e.g., to ensure a business process is executed as expected. However, these data, messages, and events usually originate from different kinds of sources, each specified by different kinds of descriptions. This variety makes it difficult to automate the detection of relevant event sources for business process monitoring. In this paper, we present a course of actions to automatically associate different event sources to event object types required for business process monitoring. In particular, in a three-step approach we determine the similarity of event sources to event object types, rank those results, and derive a mapping between their attributes. Thus, relevant event sources and their bindings to specified event object types of business processes can be automatically identified. The approach is implemented and evaluated using schema matching techniques for a specific use case that is aligned with real-world energy processes, data, messages, and events.

1 Introduction

Service-Oriented Architecture (SOA) enables flexible operations by utilizing highly-distributed and loosely-coupled applications [1]. These applications use different functions encapsulated as services by information systems that are based on the SOA paradigm. These services interact with each other by the means of messages and events [2]. In parallel, these messages and events provide information required for operations (executed by services or humans) in an organization.

Nowadays, frequent changing markets and customer requirements force companies to quickly adapt their operations to stay competitive within their environment. That is why they strive to run their operations in a process-oriented way using tools and practices from Business Process Management (BPM) to achieve their goals. BPM comprises concepts, methods, and techniques to support the design, administration, configuration, enactment, and analysis of business processes [3].

One of the current research areas of BPM is Business Process Intelligence (BPI) that comprises process analysis, monitoring, and mining [4]. During the execution of business processes – run of the process instances – services and humans are utilized to fulfill certain tasks and at the same time events can occur which represent the real world happenings and the current process state. This information about process execution is essential for analysis and improvement of business processes. It can be used for monitoring, to identify current and predict upcoming process steps by observing occurred events, and deriving process behavior from them [5,6]. However, processing events in distributed and heterogeneous environments in a semantically meaningful way requires explicit information about the structure and semantics of events. This is enabled by the framework presented by Herzberg et. al [7], but includes mostly manual steps to identify and connect the right event sources. These steps are subject for automation.

In this paper, we introduce a three-step approach that enables us to automatically identify an event source in a collection of event sources which matches a given Event Object Type (EOT) of a business process. It determines the similarity of event sources to event object types, ranks those results, and derives a mapping between their attributes. Thus, this technique establishes the basis for fine-grained business process monitoring by allowing the monitoring of an unlimited amount of events that originate from different kinds of sources and are specified by different kinds of descriptions.

The remainder of the paper is structured as follows. In Section 2, we introduce basic concepts and terms we build on, followed by the introduction of our use case from the utilities domain, in Section 3. In Section 4, we describe our approach of detecting event sources and detail the three main steps. The evaluation of the presented approach using schema matching techniques based on the introduced use case is explained in Section 5. In Section 6, related work is considered before we conclude our work in Section 7.

2 Preliminaries

In our work, we explicitly distinguish between real-world events and event objects. A *real-world event* happens in a particular *point in time* at a certain *place* in a certain *context* (cf. [8,9]). The context describes the situation in which the event has happened. Real-world events that are represented in information systems are called *event objects*. An event object is an object that represents, encodes, or records a real-world event, generally for the purpose of computer processing [10]. We embed our approach in the framework from Herzberg et. al [7] which is described below.

Real-world events are observed and published as machine-readable event objects via *event sources* (see Fig. 1). Typically, an event source publishes one specific type of event objects, e.g., the European Energy Exchange publishes the values of the European Electricity Index. The *event source description* defines the structure and semantics of event objects the event source provides.

The structure of event objects that need to be processed is described by an *EOT* (see Fig. 1). In particular, the EOT (i) defines the structure of the event content and (ii) provides the rules for mapping the information of real-world events into event objects, the so-called binding.

Definition 1 (Event Object Type (EOT))
An event object type $EOT = (cd, bind)$ consists of a specification of the event content structure cd and a binding $bind$ that describes the way how the information of a concrete event source needs to be utilized to form an event object. ◇

A binding is a function specifying the rules and methods to extract real-world event information from the event sources for an EOT. The implementation of the extraction

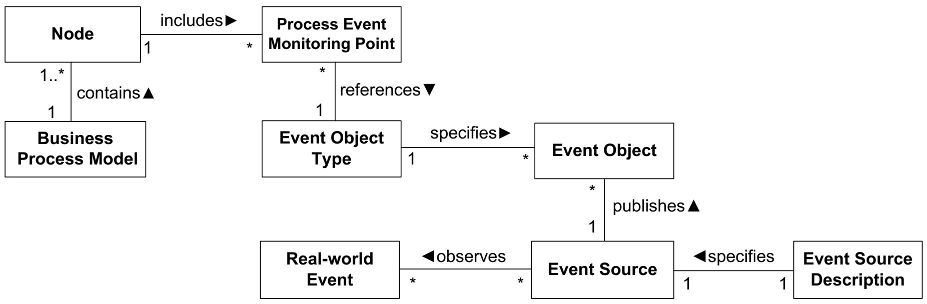

Fig. 1. The connection between a business process model and real-world events

of real-world event information can be established by applying techniques and methods from the domain of Complex Event Processing (CEP).

Event objects need to be assigned to specific points in the business process to enable BPI. Business processes are specified in business process models.

Definition 2 (Business Process Model)
A business process model $M = (N, F)$ consists of nodes N and flow relations $F \subseteq N \times N$ that connect nodes. Every node $n \in N$ has an assigned life cycle $C : N \rightarrow 2^S \times 2^T$ consisting of states S and state transitions $T \subseteq S \times S$. $\diamond$

We utilize the concept of Process Event Monitoring Points (PEMPs) that define at which point in a process an event object is expected [7] (see Fig. 1). Therefore, a PEMP references an EOT. This connection is defined during design time in a process model. The points in the process model a PEMP can be assigned to are described by life cycles of the nodes, i.e., activities, gateways, and events. Assigning PEMPs on node life cycle level enables fine-grained BPI.

Definition 3 (Process Event Monitoring Point)
Let $M = (N, F)$ be a process model and $C(n) = (S', T')$ the node life cycle consisting of states $S' \in 2^S$ and state transitions $T' \in 2^T$ for a node $n \in N$. A *process event monitoring point* is a tuple $PEMP = (M, n, t, E)$, where M is the process model it is contained in, $n \in N$ is the node it is created for, $t \in T'$ is the state transition within the node life cycle of n, and E represents the set of references to its defined EOTs. $\diamond$

3 Use Case

We introduce the approach of automating the event source detection along an example from the utilities domain. We consider the change of an energy supplier (COS) at the distribution organization from the request of a COS until all parties are informed about the COS or the rejection of the transactions. As a common technique to represent business processes, we illustrate the described process using a Business Process Model and Notation (BPMN) [11] process model shown in the upper part of Fig. 2. A customer requests a change of supplier at its new selected energy supplier. This request is then forwarded to the distributor by the new supplier and therewith, the change of supplier

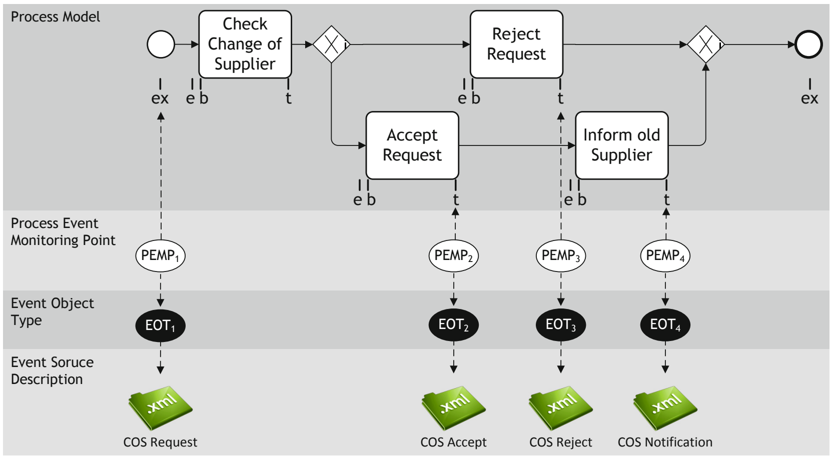

Fig. 2. Framework proposed by Herzberg et. al [7] applied to an use case from the utilities domain: business process model for handling the change of an energy supplier for a household from the distributor point of view, PEMPs associated to the node state transitions (e)nable, (b)egin, (t)erminate, or (ex)ecute, EOTs, and event sources

process at distributor site starts (start event). Afterwards the request is checked (*Check Change of Supplier*) for feasibility by the distributor and the corresponding acceptance or rejection is handled (*Accept Request* and *Reject Request*). In case the COS is accepted by the distributor a notification about the change of supplier is send to the old energy provider (*Inform old Supplier*). Afterwards, no further activities have to be performed by the client and the process ends (end event).

The data created during that process deliver valuable insights about the progress of each process instance. In our recent approach [7], we introduced a framework that allows to define PEMPs with which we are able to associate EOTs originating from different sources to business processes and enable BPI. Fig. 2 illustrates this association. Below the process model, the PEMPs are associated with state transitions of activities, start or end events, and thus contain the connection to a business process. To transform real-world events from different event sources to events relevant for the business process we define EOTs. We assume the association of an EOT to a PEMP is given and focus on identifying the binding between an EOT and an event source.

Examples for the use case are taken from real-world cases of the energy market. In particular, we used the data of the Association of Swiss Electricity Companies[1]. It provides 73 publicly available XML schema definitions (XSDs) which describe the message interchange format of Swiss electricity market[2] and represent our event sources. Their structure is rather complex, as they are interlinked with each other and contain a lot of complex element structure descriptions. For instance, the COS request schema definition (*RequestToMPA*) and its linked core component, shown in Listing 1.1,

[1] `http://www.strom.ch/`

[2] Schemas, descriptions, and examples are available at:
`http://www.strom.ch/de/dossiers/strommarkt/sdat.html`

imports 30 other schemas, defines 22 elements and 62 complex types used for the structure definition. In Line 12 of Listing 1.1, we see the structure in which the switch date period of the COS request is indicated. This element is defined by a complex type, see Line 20, which refers to another simple type, see Line 26.

Furthermore, the systems used for message exchange, as well as the content of the COS-related messages may differ for each supplier that is communicating with the distribution organization in this process. Because of that numerous amount of structure information it is difficult to manually define the binding between the event sources and the defined EOTs. Therefore, an automatic approach is necessary.

Listing 1.1. An excerpt of a COS request schema definition (RequestToMPA)

```
1  <xsd:schema xmlns:rsm=”http://www.strom.ch” ...>
2      <xsd:element name=”RequestToMPA_10” type=”rsm:RequestToMPAType_10”/>
3      <xsd:complexType name=”RequestToMPAType_10”>
4          <xsd:element name=”MeteringPoint”>
5              <xsd:complexType>
6                  <xsd:complexContent>
7                      <xsd:extension base=”rsm:EnergyMeteringPointLocationType”>
8                          <xsd:sequence>
9                              <xsd:element name=”BalanceResponsible” type=”rsm:EnergyPartyType”
                                       minOccurs=”0”/>
10                             <xsd:element name=”BalanceSupplier” type=”rsm:EnergyPartyType” minOccurs=”
                                       0”/>
11                             <xsd:element name=”Consumer” type=”rsm:ConsumerEnergyPartyType”
                                       minOccurs=”0”/>
12                             <xsd:element name=”SwitchDate” type=”rsm:SwitchDatePeriodType”/>
13                         </xsd:sequence>
14                     </xsd:extension>
15                 </xsd:complexContent>
16             </xsd:complexType>
17         </xsd:element>
18         ...
19     </xsd:complexType>
20     <xsd:complexType name=”SwitchDatePeriodType”>
21         <xsd:choice>
22             <xsd:element name=”StartDate” type=”rsm:EnergyDateType”/>
23             <xsd:element name=”EndDate” type=”rsm:EnergyDateType”/>
24         </xsd:choice>
25     </xsd:complexType>
26     <xsd:simpleType name=”EnergyDateType”>
27         <xsd:restriction base=”udt:DateType”>
28             <xsd:pattern value=”.{4}-.{2}-.{2}”/>
29         </xsd:restriction>
30     </xsd:simpleType>
31     ...
32 </xsd:schema>
```

To monitor the start of the process, we define an EOT indicating the execution of the start event, see Listing 1.2. Because of business needs, we are explicitly interested in information about a COS request (*MPA_Request*), consumer, switch date, and balance supplier.

Listing 1.2. The EOT definition to identify events for starting a change of supplier process

```
<?xml version=”1.0” encoding=”UTF-8”?>
<xs:schema xmlns:rsm=”http://www.strom.ch” xmlns:xsd=”http://www.w3.org/2001/XMLSchema”...>
    <xs:element name=”MPA_Request” type=”xs:string”></xs:element>
    <xs:element name=”Consumer” type=”xs:string”></xs:element>
    <xs:element name=”Date” type=”xs:date”></xs:element>
    <xs:element name=”BalanceSupplier” type=”xs:string”></xs:element>
</xs:schema>
```

Listing 1.3 shows an excerpt of a real-world event from the event source for the COS requests. To transform such an event to an event relevant for the given business process, we need to define a binding between EOT_1 and the event source structure RequestToMPA_10. Manually derived from the example, we can define a binding $bind$ of EOT_1 including, for example, the switch date:
EOT_1.Date = *RequestToMPA_10.Switch.MeteringPoint.SwitchDate.StartDate*.

The other elements of EOT_1 can be mapped and the binding of the other EOTs can be defined accordingly. Establishing this bindings manually is time-consuming and error-prone, therefore, we present an approach establishing the bindings automatically.

Listing 1.3. Event of event source COS request

```
<rsm:RequestToMPA_10 xsi:schemaLocation="http://www.strom.ch RequestToMPA_1p0.xsd" xmlns:rsm="http://
        www.strom.ch" xmlns:xsi="http://www.w3.org/2001/XMLSchema−instance"> ...
    <rsm:Switch>
        <rsm:DocumentID>DID_Beispiel_3_2_1_x_LiefWec_10_11c</rsm:DocumentID>
        <rsm:MeteringPoint>
            <rsm:VSENationalID schemeID="VSE" schemeAgencyID="260">
                CH9999901234500000000000000000002</rsm:VSENationalID>
            <rsm:BalanceSupplier>
                <rsm:ID><rsm:EICID schemeAgencyID="305">12X−PARTY−A−000K</rsm:EICID></
                    rsm:ID>
            </rsm:BalanceSupplier>
            <rsm:SwitchDate><rsm:StartDate>2008−06−30</rsm:StartDate></rsm:SwitchDate>
            ...
        </rsm:MeteringPoint>
    </rsm:Switch>
</rsm:RequestToMPA_10>
```

4 Approach to Detect Bindings between EOTs and Event Sources

As defined in Section 2, EOTs are defined in a PEMP for a process model. In this paper, we assume the EOTs for a business process are known and relevant event sources with its descriptions are available. Thus, we focus on the association of event sources to corresponding EOTs. We perform our approach in three steps (see Fig. 3), which are closely interrelated, but should be considered isolated as they can be implemented by different tools, algorithms, or means.

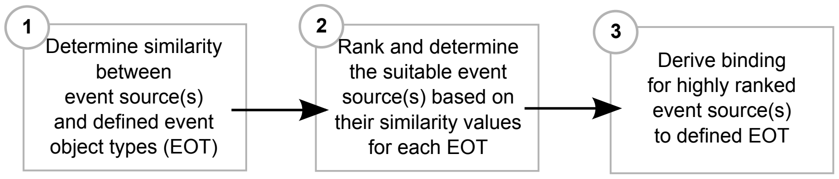
1
Determine similarity between event source(s) and defined event object types (EOT)
2
Rank and determine the suitable event source(s) based on their similarity values for each EOT
3
Derive binding for highly ranked event source(s) to defined EOT

Fig. 3. Approach for the detection of suitable event sources for given EOTs

4.1 Step 1: Determine the Similarity between Event Sources and EOTs

In a collection of event sources, e.g., systems providing XML data, we aim at identifying those event sources that are similar and suitable to a defined EOT. This identification is conducted by comparing the elements of an EOT with every event source available and computing their similarity value.

A comparison requires the description and structural properties of the events originating from an event source and those that are expected for the EOT. The structural descriptions of event sources need to be stored in a repository, e.g., a database storing the descriptions as XSD files. The search of event sources in the repository can be utilized by matchers that compare the descriptions. As we cannot assume that each event source description is completely contained in an EOT and vice versa, element-level matchers are required that consider elements in isolation and ignore element's substructure and components. For instance, the fact that the elements *Date* defined in the EOT (see Listing 1.2) and the *SwitchDate* from the event source (see Listing 1.1) are likely to match (although contained in different sub-elements) can be derived by a name-based element-level matching without considering the hierarchical structure of the descriptions. As result of a comparison, we receive a matrix for each event source that includes the similarity values between the elements of the compared event source and the EOT.

Afterwards, the similarity values in each matrix need to be aggregated to receive a global similarity value for the EOT and the event source. It is likely to estimate the degree of the similarity is given by a normalized numeric value in the range 0 to 1, in order to identify the best matching event source. For example, in our scenario, the total amount of 73 event sources must be compared with the given EOT shown in 1.2 to calculate the overall similarity values.

4.2 Step 2: Rank and Determine Suitable Event Sources for an EOT

In this step, event sources that may deliver information for the requested EOT are ranked based on their overall similarity from highest to lowest. We use the computed similarity values of each comparison of an event source description with the EOT specification from the previous step. The assumption is that higher overall similarity values indicate the better suitability of the event sources to the EOT. However, note that these values do not contain an evaluation of how suitable the mappings of each attribute are. Therefore, it may be required that a domain expert looks into the set of event sources ranked at the first places to exactly evaluate their fitness. As we are aiming at automating the process of event detection, we use the event source ranked at the first place for further processing. According to our scenario, we rank all 73 event sources based on their similarity to the requested EOT shown in Listing 1.2. Such a ranking should indicate that the event source *RequestToMPA_10* will satisfy the given EOT most probably.

4.3 Step 3: Derive the Binding between an EOT and an Event Source

For the highly ranked event source(s) the binding is derived, so that for the requested elements or attributes in the EOT a corresponding counterpart from the event source(s) is assigned. The binding defines the rules and methods to extract raw event information from the data source within the information system landscape required for the events of the given EOT. To establish the binding the resulting similarity matrix from *step 1* can be reused. Same as for the global similarity, we assume that the higher similarity value are better suited and thus they are used for the matching of elements. Therefore, each required binding is produced by utilizing the element in the event source with the most promising similarity value. For example, it is likely to assume that the similarity value between the elements *Date* (of the EOT) and *SwitchDate* (of the event source) is higher then a value between *Date* and *Consumer*.

Based on the similarity matrix, we can obtain a binding of every element described by the EOT to an element in the determined event source. Using this technique, we can derive a binding between the EOT_1 requested in the scenario and the event source *RequestToMPA_10* describing that, for example, the content for filling *Date* can be found in *R2MPAType_10.Switch.MeteringPoint.SwitchDate*. This binding can then be used to generate CEP-queries and process events in a CEP engine or to generate XPath-like pattern and process XML documents using Extensible Stylesheet Language Transformation (XSLT), for instance. Both can be used to detect corresponding events and data coming from different sources but belonging to the specified EOT. In this context, the existing Event Processing Platform (EPP) [6,7][3] can be used as it is able to receive events from different kinds of sources, check the format of these events, associate them with an EOT, and then normalize, store, process, and forward them via the platform.

5 Evaluation Using Schema Matching

We show the applicability of our approach by using schema matching to implement the three steps and evaluate it for the use case described in Section 3. By automating the event source detection, we can decrease the efforts for enabling process monitoring and analysis in distributed and heterogeneous environments. In this evaluation, we did not investigate the quality of the matching results.

In general, schema matching techniques are used to identify an alignment that expresses the correspondence between elements and attributes belonging to different schemas or ontologies [12,13]. A schema can be, for example, an XSD, a database schema, an Electronic Data Interchange For Administration, Commerce and Transport (EDIFACT) message definition, or an event type definition. To apply schema matching, we assume that each event source as well as the EOT are represented by a schema, i.e., an XSD that describes the events produced by that event source as well as the expected events in a formal and standardized manner. This assumption is reasonable since XML is a common standardized data interchange format.

5.1 Evaluation Setup

For our experiment, we used a state-of-the-art schema matching tool called OntoBuilder[4] to implement the identification and matching of suitable event sources for EOTs. For the comparison in the OntoBuilder, we first defined an EOT in form of an XSD for each PEMP required by the use case in Section 3. The related XSDs of the event sources are taken from the data set provided by the Association of Swiss Electricity Companies (see Section 3). Second, each XSD was imported in the OntoBuilder and transformed to a generic ontology structure. Third, we conducted a schema matching for each pair of a target (EOT) and source schema (event source).

[3] Downloads, tutorials, and further information can be found at:
`http://bpt.hpi.uni-potsdam.de/Public/EPP`

[4] See `http://ontobuilder.bitbucket.org/`

Table 1. Excerpt of results for matching of EOT_1 and event sources

#	Event source schema	Average similarity
1	RequestToMPA_1p0.xsd	0.78
2	ResponseFromMPA_Confirm_1p0.xsd	0.71
3	NotificationFromMPA_1p0.xsd	0.67

In the OntoBuilder, the schema matching is based on a sequence of two kinds of matchers: (1) first line matchers and (2) second line matchers. The first line matchers are applied to two schemas to determine their similarity using textual and structural heuristics. For each schema pair a first line matcher produces a similarity matrix which contains the similarity value for each attribute pair of those two schemas. In our use case each similarity value is calculated by comparing the names as well as the data types of the schema attributes. For this reason, we used the *OBTermMatch* algorithm for the name comparison and the *OBValueMatch* algorithm for the data type comparison. Then, a second line matcher is applied to the similarity matrix together with a set of constraints (e.g., allowing only 1:1 mappings of attributes) that returns for each attribute of the target schema a corresponding attribute of the source schema together with its computed similarity value. *OBStableMariage* was used as a second line matcher that implements the stable marriage algorithm for royal couples [14], extended for unequal sets of schema attributes.

Following this setup, our approach uses schema matching in the OntoBuilder as follows: for *step 1* the overall similarity of the EOT to an event source is calculated as the arithmetic mean of their similarity values, for *step 2* the overall similarity values of the event sources for each EOT are ranked, and for *step 3* the binding is derived from the matching between the attributes of event sources to attributes of the EOT.

5.2 Evaluation Results

The results presented in Table 1 show an excerpt of found event source descriptions that match the event type EOT_1 (see Listing 1.2) ordered by their computed overall similarity. The event source description *RequestToMPA_1p0.xsd*, which represents the event source *RequestToMPA_10* (see Listing 1.1), is ranked with the highest similarity of 0.78. It describes the supplier change request to a metering point administrator within Association of Swiss Electricity Companies and corresponds to the definition of the EOT_1 (see Listing 1.2).

Next, the matching of attributes were computed in the OntoBuilder. For example, using the *OBTermMatch* algorithm, which includes a set of weighted string comparison algorithms for similarity calculation, we receive a similarity between *Date* (EOT) and *SwitchDate* (event source) of 0.46, while the similarity between *Consumer* (EOT) and *Consumer* (event source) is 1. Table 2 shows the results of the mappings for each attribute of the EOT_1 to the attributes of the best matched event source description (*RequestToMPA_1p0.xsd*) and the similarity values for each mapping.

Table 2. Results for matching the attributes of EOT_1 and the best ranked event source description *RequestToMPA_1p0.xsd*

EOT_1 attribute	**RequestToMPA_1p0.xsd attribute**	**sim. value**
MPA_Request	RequestToMPA_10	0.65
Consumer	R2MPAType_10.Switch.MeteringPoint.Consumer	1.00
Date	R2MPAType_10.Switch.MeteringPoint.SwitchDate	0.46
BalanceSupplier	R2MPAType_10.Switch.MeteringPoint.BalanceSupplier	1.00

5.3 Evaluation Discussion

The produced results show that schema matching can be used for the identification of event sources in our introduced use case. The approach allows to identify heterogeneous event sources for a given EOT and to derive bindings for each of event sources. The derived bindings can, for example, be used to generate and run CEP-queries.

We argue that schema matching is a reasonable technique which can be used for an automated detection of event sources. However, the quality of the produced results may highly depend on the selected matching algorithms as well as on their configuration settings and may vary with the examined data set. Furthermore, the selection and adaption of first and second line matcher algorithms have a high impact on the outcome of the produced matching. Additional element-level matchings such as on type, description, and key property can be applied as well (see [12]).

In this paper, we focus on the binding of event sources to EOTs. Although the combination and association of several varying EOTs to one PEMP as well as the combination and association of several varying event sources to one EOT is possible. While illustrating our approach, we did not analyze these details, however, we plan to consider them in future work.

6 Related Work

Baier and Mendling introduce an approach to map events from event logs to activities in process models [15,16]. The mapping approach suggests relations between events and activities at type and instance level in an automated manner. These relations are derived from a comparison between the name of event types and activity's name and description. We complement this approach by considering all attributes of EOTs and of the event sources to identify a matching between these.

Other approaches in the area of complex event processing e.g., [17] address the problem of semantic decoupling of event subscriber and publisher in heterogeneous event-based systems. For instance, the approach of Hasan et al. includes an event model, a subscription model, and a matching model that leverage semantics of events and subscriptions (resp. EOT) to establish approximate matching.

Our work is also related to the area on automated or semi-automated techniques for schema matching, surveyed, for instance, in [12] and [18]. These techniques leverage textual and structural similarity measures, and apply different strategies to derive correspondences between concepts from such a similarity assessment [19]. While the results of schema matchers have been exceptional for distinguished settings, they are inherently uncertain due to the enormous ambiguity and heterogeneity of data description [20].

Several approaches were proposed to address the problem of service retrieval - a process of finding a set of service candidates for a given consumer request. The lack of an appropriate service description can be identified as one of the main challenges during the process of service retrieval [21]. Various approaches, e.g., [22], address these challenges using diverse information retrieval techniques. While these approaches allow the service consumer to reach high levels of recall, they remain to be prone to low precision [23].

A few works use schema matching techniques in service retrieval. In [24], the authors propose an approach to compute the similarity between web services under consideration of a tree edit distance. However, the introduced approach can be applied on the interface definition of web services only. It does not consider the exchanged data and their instances as it is needed for event source detection.

7 Conclusion

We automated the association of event sources to EOTs required by PEMPs in a business process and therewith enabled event source detection as basis for BPI. In particular, specified EOTs for business process monitoring are used to identify relevant event sources. The found event sources are ranked according their computed similarity to the requested EOT and a binding between the highest ranked event source and the EOT is generated. During business process execution, these bindings can be used to identify event objects for a business process in an event stream. We implemented the approach using schema matching and evaluated its applicability for an use case that is aligned with a real-world process from the utilities domain.

For future work, we plan to consider the combination and association of several varying EOTs to one PEMP as well as the combination and association of several varying event sources to one EOT. Furthermore, we aim to extend our evaluation using different kinds of configurations and combinations of matching algorithms and, thus, improve the quality of event source detection.

Acknowledgement. The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement 318275 (GET Service) and from the German Research Association (DFG) under project number WE 1930/8-1.

References

1. Papazoglou, M.P., Heuvel, W.J.: Service oriented architectures: approaches, technologies and research issues. The VLDB Journal 16(3) (July 2007)
2. Levina, O., Stantchev, V.: Realizing Event-Driven SOA. In: 4th International Conference on Internet and Web Applications and Services, ICIW (2009)
3. Weske, M.: Business Process Management - Concepts, Languages, Architectures, 2nd edn. Springer (2012)

4. van der Aalst, W.: Process Mining: Overview and Opportunities. ACM Transactions on Management Information Systems (TMIS) 3(2) (July 2012)
5. Weidlich, M., Ziekow, H., Mendling, J., Günther, O., Weske, M., Desai, N.: Event-Based Monitoring of Process Execution Violations. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 182–198. Springer, Heidelberg (2011)
6. Bülow, S., Backmann, M., Herzberg, N., Hille, T., Meyer, A., Ulm, B., Wong, T.Y., Weske, M.: Monitoring of Business Processes with Complex Event Processing. In: BPM Workshops. Springer (2013) (accepted for publication)
7. Herzberg, N., Meyer, A., Weske, M.: An Event Processing Platform for Business Process Management. In: IEEE International EDOC Conference, Vancouver (2013)
8. Etzion, O., Niblett, P.: Event Processing in Action. Manning Publications Co (2011)
9. Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley (2002)
10. Luckham, D., Schulte, R.: Event Processing Glossary - Version 2.0 (July 2011), `http://www.complexevents.com/wp-content/uploads/2011/08/EPTS_Event_Processing_Glossary_v2.pdf`
11. OMG: Business Process Model and Notation (BPMN), Version 2.0 (2011), `http://www.omg.org/spec/BPMN/2.0/`
12. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. The VLDB Journal 10(4) (2001)
13. Euzenat, J., Shvaiko, P.: Ontology matching, vol. 18. Springer, Heidelberg (2007)
14. Marie, A., Gal, A.: On the Stable Marriage of Maximum Weight Royal Couples. In: AAAI Workshop on Information Integration on the Web (2007)
15. 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.) BPM 2013. LNCS, vol. 8094, pp. 17–32. Springer, Heidelberg (2013)
16. Baier, T., Mendling, J.: Bridging Abstraction Layers in Process Mining: Event to Activity Mapping. In: Nurcan, S., Proper, H.A., Soffer, P., Krogstie, J., Schmidt, R., Halpin, T., Bider, I. (eds.) BPMDS 2013 and EMMSAD 2013. LNBIP, vol. 147, pp. 109–123. Springer, Heidelberg (2013)
17. Hasan, S., O'Riain, S., Curry, E.: Approximate semantic matching of heterogeneous events. In: 6th ACM International Conference on Distributed Event-Based Systems, DEBS (2012)
18. Bellahsene, Z., Bonifati, A., Rahm, E.: Schema Matching and Mapping. Springer (2011)
19. Gal, A., Sagi, T.: Tuning the ensemble selection process of schema matchers. Information Systems 35(8) (2010)
20. Gal, A.: Managing Uncertainty in Schema Matching with Top-K Schema Mappings. In: Spaccapietra, S., Aberer, K., Cudré-Mauroux, P. (eds.) Journal on Data Semantics VI. LNCS, vol. 4090, pp. 90–114. Springer, Heidelberg (2006)
21. Kuropka, D., Troeger, P., Staab, S., Weske, M.: Semantic Service Provisioning. Springer (2008)
22. Wang, Y., Stroulia, E.: Flexible Interface Matching for Web-Service Discovery. In: 4th Int. Conference on Web Information Systems Engineering (WISE). IEEE Computer Society (2003)
23. Klein, M., Bernstein, A.: Toward high-precision service retrieval. IEEE Internet Computing 8(1), 30–36 (2004)
24. Hao, Y., Zhang, Y.: Web services discovery based on schema matching. In: 13th Australasian Conference on Computer Science (ACSC), vol. 62, pp. 107–113. Australian Computer Society, Inc. (2007)