Ontology-Based Data Access for Extracting Event Logs from Legacy Data: The onprom Tool and Methodology

Diego Calvanese[1], Tahir Emre Kalayci[1(✉)], Marco Montali[1], and Stefano Tinella[2]

[1] KRDB Research Centre for Knowledge and Data, Free University of Bozen-Bolzano, Bolzano, Italy
{calvanese,tkalayci,montali}@inf.unibz.it
[2] EBITmax srl, via Macello 63/F, Bolzano, Italy
s.tinella@ebitmax.it

Abstract. Process mining aims at discovering, monitoring, and improving business processes by extracting knowledge from event logs. In this respect, process mining can be applied only if there are proper event logs that are compatible with accepted standards, such as extensible event stream (XES). Unfortunately, in many real world set-ups, such event logs are not explicitly given, but instead are implicitly represented in legacy information systems. In this work, we exploit a framework and associated methodology for the extraction of XES event logs from relational data sources that we have recently introduced. Our approach is based on describing logs by means of suitable annotations of a conceptual model of the available data, and builds on the ontology-based data access (OBDA) paradigm for the actual log extraction. Making use of a real-world case study in the services domain, we compare our novel approach with a more traditional extract-transform-load based one, and are able to illustrate its added value. We also present a set of tools that we have developed and that support the OBDA-based log extraction framework. The tools are integrated as plugins of the ProM process mining suite.

Keywords: Process mining · Ontology-based data access · Event log extraction · Relational database management systems

1 Introduction

Contemporary organizations are increasingly recognizing the importance of analyzing how their business processes are conducted in the real world, towards quality assurance, optimization, and continuous improvement. Process mining [1] is emerging as one of the most promising and effective framework to tackle this need. Process mining stands at the intersection of model-driven engineering and data science: insights are automatically extracted from event data that represent the footprint of process executions inside the company, and used to discover and enrich process models, provide operational support, check compliance, analyze bottlenecks, compare process variants, and suggest improvements [2]. A plethora of process mining techniques and technologies have been developed and successfully employed in several application domains[1].

The applicability of process mining depends on two crucial factors:

– the availability of high-quality event data, that is, logs containing correct and complete data about which cases (process instances) have been executed, which events occurred for each case, and when they did occur;
– the representation of such data in a format that is understandable by process mining algorithms, such as the IEEE XML-based standard eXtensible Event Stream (XES) [3].

In this respect, two main situations typically arise in an industrial setting. In the first situation, the company explicitly adopts a business process or enterprise management system that logs cases, events and corresponding attributes explicitly, facilitating the extraction of an event log and its conversion into XES.

The literature abounds of techniques and tools to handle the log extraction in this setting, such as, e.g., XESame [4] and ProMimport [5]. Additionally, commercial tools like Disco[2], Celonis[3], and Minit[4] support the conversion from CSV or spreadsheet files into XES. Worth mentioning are also [6,7], the first because it tackles the extraction of event logs from redo-logs of relational databases, the second because it is one of the few approaches that leverages the relational technology to access the event log directly, instead of materializing it into XML.

In the second situation, the company adopts a more general management system, configuring it for its own specific needs, and combining it with domain-specific databases and other legacy data sources. In this setting, cases and events may not be explicitly stored in dedicated data structures, but instead implicitly present inside the company information system. In addition, there is typically not a single notion of "case" and related "events", but they change depending on the perspective of interest, and on which aspects of the company one wants to focus on. For example, in an order-to-cash process, one could focus on the flow of orders, to understand why sometimes orders take too much time to be delivered, or on the flow of operations conducted by a warehouse employee, to check whether it complies with internal regulations. Depending on which notion of case is selected, also the relevant events change. E.g., the payment of an order is important when analyzing the flow of orders, but may be irrelevant when focusing on the warehouse.

Unfortunately, the literature lacks techniques, methodologies and tools to support domain experts and process analysts in the extraction of event logs from legacy information systems, and reflecting multiple perspectives. The result is that logs are extracted manually, adopting ad-hoc procedures that are based on extracting a copy of the data and transforming it according to specific requirements. This process, which resembles the extract-transform-load (ETL) approach taken for data warehousing, creates redundancy, and is labor intensive and error prone.

[1] http://tinyurl.com/ovedwx4.
[2] https://fluxicon.com/disco/.
[3] http://www.celonis.de/en/.
[4] http://www.minitlabs.com.

In this paper, we tackle this open challenge. Leveraging the technique first presented in [8], we propose an approach based on conceptual modeling to semi-automatize the extraction of event logs from legacy information systems. In our approach, called onprom, humans only focus on the conceptual issues involved in the extraction: *(i)* Which are relevant concepts and relations? *(ii)* How do such concepts/relations map to the underlying information system? *(iii)* Which concepts/relations relate to the notion of case, event, and event attributes?

Once this information is provided, the log extraction process is handled in a fully automatized way, leveraging the paradigm of *ontology-based data access* (OBDA) [9–11]. In OBDA, a high-level representation of the domain of interest, in our case provided in terms of a conceptual schema, is linked to the data sources using a declarative specification, called *mappings*. In this way, information about the event logs can be extracted from the sources by exploiting both the conceptual schema and the mappings.

In the following, we describe a real process mining use case that has been initially handled using an ad-hoc, ETL-like methodology. Employing this use case as a running example, we then introduce our onprom methodology, discussing its different phases and how conceptual models are used both as documentation and computational artifacts. We then show how the preliminary implementation reported in [8] has now been transformed into a complete chain of tools, fully integrated with the well-known ProM process mining framework.

2 Case Study and Motivation

To provide a concrete motivation and explanation for our framework, we introduce the problem of extracting event logs from legacy information systems in a real case study. The case study has been carried out by EBITmax[5], an innovative SME from South Tyrol, Italy. EBITmax provides consultancy services in program management and business process management for a number of small and large enterprises, operating within the territory and abroad. Recently, EBITmax incorporated process mining to complement its standard consultancy services, enriching and comparing models with fine-grained insights automatically extracted from data, and accounting for how business processes are executed in reality. In particular, a pilot project in process mining is currently run by EBITmax on the service provisioning and financial processes of Markas[6], a company with more than 7 000 employees providing a multitude of services for large establishments operating in Italy, Austria, and Romania. Specifically, the pilot consists in the analysis of the *Accounts Payable Process* (App), used by Markas to handle payments to external suppliers, and their corresponding invoices. To support the internal management of the App, Markas does not employ a workflow management system, but relies on shared guidelines on how to handle payments, and on an *Enterprise Resource Planning* (ERP) system to track the executed operations. In this setting, Markas management would like to understand whether the

[5] http://www.ebitmax.it.
[6] http://www.markas.com/en/home.html.

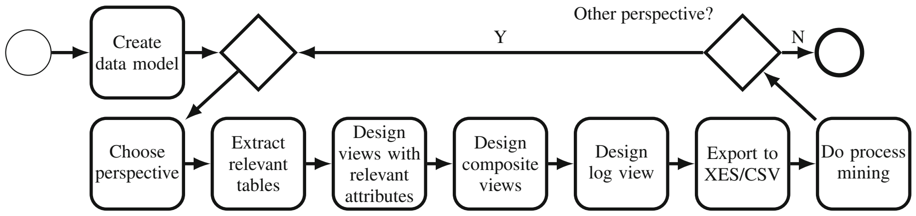


Fig. 1. Traditional methodology for data preparation for process mining

App is executed as expected and, if not, where do deviations appear, considering all the orders created in 2015.

2.1 Understanding the Problem

The first step followed by EBITmax has been to understand the details of the App, both conceptually and in terms of IT support. On the one hand, this resulted in the creation of a model for the expected App that is expressed in the *Business Process Model and Notation*[7] (BPMN) standard. The model has been obtained following a traditional interview-based approach. On the other hand, this resulted in the annotation of the BPMN model, so as to know which tasks are executed manually, and which are performed through the ERP system, and are consequently logged. Among the logged tasks, we mention the following, key ones:

- SubmitOrder: an order is inserted into the ERP and submitted to a supplier.
- GetTD: Markas receives the ordered material; this is traced in the ERP when the transport document (TD) attached to the material is inserted into the system.
- RegisterInvoice: the invoice for the payment is inserted into the ERP, reflecting what is listed in the TD.
- PaySupplier: the payment is confirmed.

Normally, the management expects that *the invoice of an order is not registered (and, consequently, not paid) unless the official TD is obtained and inserted into the ERP*. Within the pilot project, the general question of the alignment between the expected and the actual App boiled down to check whether the business constraint "no invoice unless TD" is indeed respected by the actual App.

2.2 Process Mining via Manual Event Log Extraction

To answer the research question through process mining, EBITmax needed to tackle the difficult issue of data preparation, which is considered one of the most challenging, open problems in process mining [1]. In this specific case, the

[7] http://www.bpmn.org/.

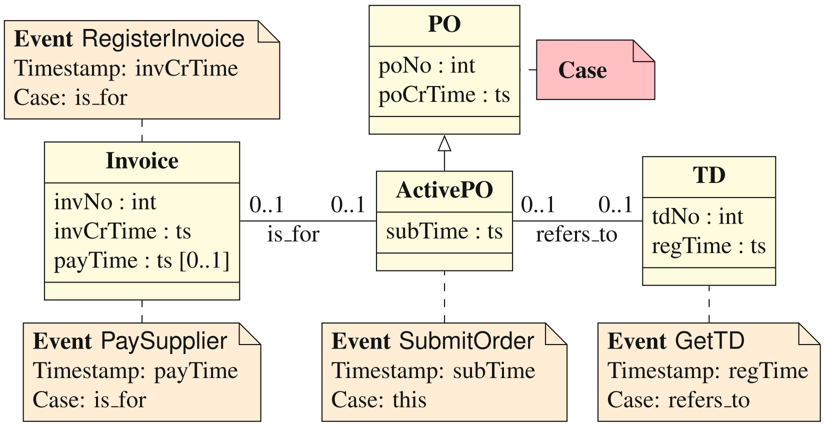


Fig. 2. Excerpt of the data model for App

concrete problem was to identify the semantics of tables/columns of the ERP, and the whereabouts of relevant data for orders, TDs, invoices, and payments. To this end, EBITmax devised and documented the methodology shown in Fig. 1.

Conceptual data modeling. The first step of the methodology consists in the creation of a conceptual model that accounts for the data maintained in the ERP at a higher level of abstraction, on the one hand making it possible to discuss with managers and domain experts about the semantics of such data, and on the other hand providing the basis to understand where and how they are stored within the ERP. The UML class diagram in Fig. 2 depicts a small excerpt of the resulting model, showing the key concepts of PO (purchase order), TD (transport document), and Invoice. ActivePO represents a PO that has been submitted, consequently triggering the execution of an App instance.

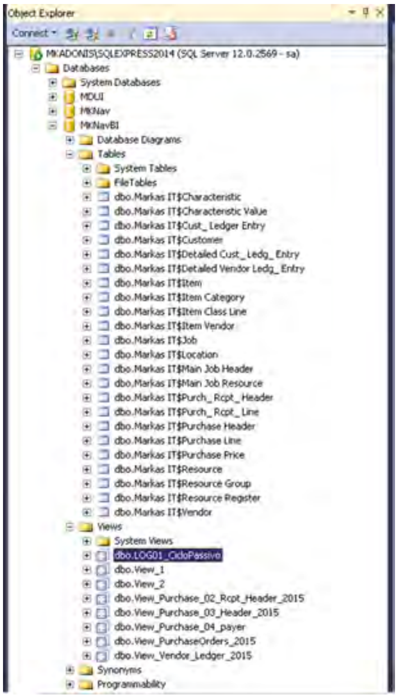

```
Object Explorer
Connect
MKADONIS\SQLEXPRESS2014 (SQL Server 12.0.2569 - sa)
  Databases
    System Databases
    MDUI
    MKNav
    MKNavBI
      Database Diagrams
      Tables
        System Tables
        FileTables
        dbo.Markas IT$Characteristic
        dbo.Markas IT$Characteristic Value
        dbo.Markas IT$Cust_ Ledger Entry
        dbo.Markas IT$Customer
        dbo.Markas IT$Detailed Cust_Ledg_ Entry
        dbo.Markas IT$Detailed Vendor Ledg_ Entry
        dbo.Markas IT$Item
        dbo.Markas IT$Item Category
        dbo.Markas IT$Item Class Line
        dbo.Markas IT$Item Vendor
        dbo.Markas IT$Job
        dbo.Markas IT$Location
        dbo.Markas IT$Main Job Header
        dbo.Markas IT$Main Job Resource
        dbo.Markas IT$Purch_ Rcpt_ Header
        dbo.Markas IT$Purch_ Rcpt_ Line
        dbo.Markas IT$Purchase Header
        dbo.Markas IT$Purchase Line
        dbo.Markas IT$Purchase Price
        dbo.Markas IT$Resource
        dbo.Markas IT$Resource Group
        dbo.Markas IT$Resource Register
        dbo.Markas IT$Vendor
      Views
        System Views
        dbo.LOG01_CicloPassivo
        dbo.View_1
        dbo.View_2
        dbo.View_Purchase_02_Rcpt_Header_2015
        dbo.View_Purchase_03_Header_2015
        dbo.View_Purchase_04_payer
        dbo.View_PurchaseOrders_2015
        dbo.View_Vendor_Ledger_2015
      Synonyms
      Programmability
```

Fig. 3. Manual construction of views

Choosing perspective. The second step consists in combining the research question with the data model, so as to choose a *perspective* for the analysis, and in particular: *(i)* the "subject" of the analysis, i.e., which notion of case to adopt; *(ii)* which relevant events should be considered in the evolution of cases; and *(iii)* which event attributes should be included.

Manual data preparation. Building on this guideline, EBITmax started a fine-grained analysis of the ERP system and its underlying database, towards the extraction of the desired information. The extraction is executed manually in an ETL-like fashion, and is organized in four incremental phases: *(i)* identification of the relevant tables; *(ii)* creation of "filter views" that maintain, and clean up, the relevant information present in such tables (e.g., selection of interesting attributes for purchase orders); *(iii)* merging of filter views into "composite views" that provide a higher level of abstraction, and group together data belonging to the same conceptual classes/relations (e.g., all data referring to a transport document); and *(iv)* creation of a single "log view" that coherently rearranges the information present into composite views in accordance with the chosen perspective for case, events, and attributes. Figure 3 shows the selected tables and views constructed by EBITmax on top of Markas ERP.

2.3 Log Extraction and Process Mining

Finally, EBITmax converted the "log view" into a CSV file, and exploited the Disco process mining toolkit[8] for the analysis. Some interesting deviations departing from the expected App were consequently detected, and discussed with the Markas management. One of the most interesting, and quite common, deviations was represented by orders that are submitted and paid without registering the transport document at all. Looking at the data, Markas realized that this deviation is due to a recent "drift" in the management of the App, caused by the introduction of digital invoices in the Italian market. In fact, at some point suppliers equipped with e-invoicing started to digitally send the billing information related to submitted purchase orders. The information contained in the e-invoice mirrors all the TD data needed to execute the payment, and obviously is received by Markas before the ordered material and the corresponding TD. This allows one to speed-up the process, by paying as soon as the e-invoice is received. When the payment is executed, the ERP forbids further changes to the order and its related information, thus making it impossible to register the TD once it is received. Markas appreciated the findings obtained from the pilot, and recently accepted to continue the project at a bigger scale.

Experienced Issues. In spite of the promising results obtained by EBITmax during the pilot project, the company experienced several issues caused by the manual data preparation for process mining. First of all, the manual creation of views requires a detailed knowledge of the ERP tables, and is a demanding and error-prone task. In addition, whenever the perspective of the analysis is changed, it is necessary to go through all the data preparation phases again, since there is no guarantee that the selected tables, and designed views, will also be useful in relation with the new perspective. This contrasts with the process mining best practices in two respects: quality assurance of the input event log, and feasibility of quickly going through several batches of analysis by changing perspective on the company's data.

[8] https://fluxicon.com/disco/.

This experience has spawn a collaboration between EBITmax and the Free University of Bozen-Bolzano, so as to face the next phase of the Markas project in a more systematic way, starting from the log extraction approach first introduced in [8]. In particular, the ongoing collaboration is centered around the methodology and tool support described in the remainder of the paper.

3 The onprom Methodology

We now present our methodology for the semi-automated extraction of logs from legacy information systems, starting from the seminal ideas proposed in [8]. The methodology is backed up by the chain of tools described in Sect. 4.

As a starting point, we assume the existence of a legacy information system $\mathcal{I} = \langle \mathcal{R}, \mathcal{D} \rangle$, with schema $\mathcal{R}$ and set $\mathcal{D}$ of facts about the domain of interest. In the typical case where the information system is a relational database, $\mathcal{R}$ accounts for the schema of the tables and their columns, and $\mathcal{D}$ is a set of data structured according to such tables. On top of $\mathcal{I}$, our methodology is centered on the usage of conceptual models in two respects. First, they are used as documentation artifacts that explicitly capture not only knowledge about the domain of interest, but also how legacy information systems relate to that knowledge. This facilitates understanding and interaction among human stakeholders. Second, conceptual models are used as computational artifacts, that is, to automatize the extraction process as much as possible.

The overall methodology is illustrated in Fig. 4. We review the different phases next, leveraging the App case study illustrated in Sect. 2.

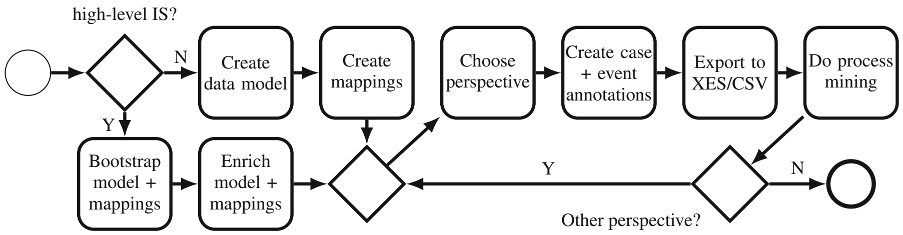


Fig. 4. The onprom methodology

3.1 Conceptual Modeling

The first phase of our methodology consists in the creation of two conceptual models. The first one is the *conceptual data model* $\mathcal{T}$, already discussed in Sect. 2.2. It accounts for the structural knowledge of the domain of interest, i.e., relevant concepts and relations, consequently providing a high-level view of $\mathcal{I}$ that is closer to domain experts. More specifically, we employ UML class diagrams as a concrete language for conceptual data modeling, and we provide their logic-based, formal encoding in terms of the OWL2 QL ontology language. OWL2 QL is one of the profiles[9] [12] of the W3C standard Semantic Web language OWL2, and it has been specifically designed to capture the essential features of conceptual modeling formalisms (see, e.g., [10]). In the following, depending on the context, we refer to $\mathcal{T}$ as a UML class diagram or as the corresponding OWL2 QL ontology.

The second conceptual model, the *mapping specification* $\mathcal{M}$, is a distinctive feature introduced by our approach, borrowed from the area of ontology-based data access (OBDA) [9,10]. $\mathcal{M}$, which explicitly links $\mathcal{I}$ to $\mathcal{T}$, consists of a set of logical implications that map patterns of data over schema $\mathcal{R}$ to high-level facts over $\mathcal{T}$. Patterns over the data $\mathcal{D}$ are expressed as queries over $\mathcal{R}$ (e.g., SQL **SELECT** statements, when $\mathcal{R}$ is relational), while facts over $\mathcal{T}$ are expressed as logical terms involving objects.

Example 1. Consider again the APP case study. In the Markas ERP, `Markas-Purchase` table contains information about purchase orders, including the primary key `No` to store the order number, and columns `Order_Date` and `Posting_Date` to store the dates at which the order was respectively created and submitted. In addition, by interacting with the domain experts of Markas, EBITmax discovered the following two important facts about such a table:

- Sometimes when the order is created, the `Order_Date` field is left unspecified[10]; however, it is still possible to reconstruct the orders created in 2015 (i.e., the year targeted by the analysis), as those whose order number starts with 15.
- When the order is created, its posting date is left unspecified, and gets a value when the order is actually submitted.

This knowledge can be made explicit by establishing dedicated mappings. In particular, the following mapping, specified using the mapping syntax of the ontop OBDA framework[11] [11]:

```
order/{oid} poNo {oid} .
    ← SELECT No AS oid FROM Markas-Purchase WHERE No LIKE '15%'
```

declares that each value *oid* that is stored in the `No` column of `Markas-Purchase` and that begins with number 15, corresponds to an object term `order/oid` in the ontology[12], and determines relationship of type `poNo` between such object and the value *oid*. Since relationship `poNo` is actually an integer attribute[13] of

[9] In W3C terminology, a *profile* is a sublanguage.

[10] It is important to notice that the possible absence of an actual value for `Order_Date` does not contrast with the class diagram of Fig. 2, which dictates that every purchase order has exactly one creation time. In fact, conceptual models are interpreted under *incomplete information*: the absence of the creation date for an order does not mean that the order has no creation date, but that such an order has a creation date that is not certainly known.

[11] http://ontop.inf.unibz.it/.

[12] In the left-hand side of a mapping, curly brackets are used to denote answer variables of the SQL query in the right-hand side.

[13] In OWL terms, it is a *data property*.

class PO in $\mathcal{T}$, this mapping is implicitly declaring also that order/*oid* belongs to class PO, and that its order number is *oid*. An example of mapping explicitly populating a class is:

```
order/{oid} rdf:type ActivePO.
  ←  SELECT No AS oid FROM Markas-Purchase
     WHERE No LIKE '15%' AND Posting_Date IS NOT NULL
```

This mapping expresses that each order tuple in Markas-Purchase identified by value *oid* is mapped to an object order/*oid* of type ActivePO in $\mathcal{T}$, whenever its posting date has a non-null value. ■

When $\mathcal{M}$ is fully defined, it can be used for two purposes. On the one hand, it explicitly documents how the structure of the company information system has to be conceptually understood in terms of domain concepts and relations, and thus constitutes an asset for the company that itself might be worth an investment [13]. On the other hand, $\langle \mathcal{I}, \mathcal{T}, \mathcal{M} \rangle$ constitutes what is called an *OBDA system*, which completely decouples end users from the details of the information system: whenever a user poses a conceptual query Q (e.g., expressed using the semantic web query language SPARQL) over $\mathcal{T}$, the OBDA system *(i)* leverages $\mathcal{T}$ and $\mathcal{M}$ to automatically reformulate Q as a corresponding concrete query Q' over $\mathcal{I}$; *(ii)* submits Q' to $\mathcal{I}$; and *(iii)* automatically translates the so-obtained answers into meaningful answers over the vocabulary of $\mathcal{T}$. Notably, this "virtual" approach is conceptually identical to the one in which the mapping $\mathcal{M}$ is used a là ETL to materialize data from $\mathcal{D}$ as facts over $\mathcal{T}$, with the advantage that: *(i)* users do not need to code procedures for data extraction, *(ii)* data are not replicated, and *(iii)* data are retrieved using the standard query engine of the information system.

Bootstrapping. The creation of a suitable data model and mapping specification is a labor-intensive and challenging task. As shown in Fig. 4, if the information system has a "high-level" structure, that is, a structure that is understandable by domain experts, such a phase can be (partially) automatized through bootstrapping techniques [14], which synthesize a conceptual data model that mirrors the structure of the information system, together with suitable mappings. The result of bootstrapping can then be manually improved and enriched towards the creation of the final OBDA system.

3.2 Event Data Annotations

Once the OBDA system is set up in the previous phase, our methodology allows one to abstract away the information system. In this way, the process mining expert is not required to manually construct views for the extraction of an event log, as done in Fig. 1. Instead, she focuses on $\mathcal{T}$ only, and uses it as the basis for discussion with the company stakeholders, in particular to decide which perspective for process mining to consider. Concretely, choosing a perspective amounts to *annotate* $\mathcal{T}$ with a set $\mathcal{L}$ of event data annotations, where each annotation is used either to: *(i)* indicate which class in $\mathcal{T}$ (possibly with additional restrictions) represents a *case*, *(ii)* which events are present in $\mathcal{T}$ and to which classes they refer, *(iii)* which attributes are attached to events, and where they are located in $\mathcal{T}$. We consider each type of annotation next.

Case annotation. The case annotation specifies which class constitutes the reference point for the analysis. Each object instantiating the case class represents an instance of the process according to the chosen perspective, and provides the basis for correlating events: the set of all events referring to the same case object form a *trace* for such an object. In Fig. 2, the case annotation is shown as a red UML note, and indicates that each purchase order is a case. Additional restrictions on which instances to consider can be applied. E.g., one could specify to consider only orders referring to suppliers from a given geographical area, or orders involving at least a given amount of money.

Event annotations. An event annotation specifies that the annotated class provides information about the occurrence(s) of a type of event that is relevant for the chosen perspective. To "discover" which classes in $\mathcal{T}$ may be subject to an event annotation, our methodology combines two constraints:

– Each event class has to be directly or indirectly linked to the case class. Technically, there must be a UML association, or a chain of concatenated associations (possibly involving IS-A generalizations), that lead to navigate from the event class to the case class.
– Each event class has to be directly or indirectly linked to a timestamp attribute, providing the information on "when" instances of such an event occurred. Technically, there must be a *functional* attribute of the event class, or a chain of one or more *functional* associations (possibly involving IS-A generalizations), that lead to navigate from the event class to its timestamp attribute.

Two observations are in place regarding the aforementioned navigations and the multiplicities attached to the involved associations. For event-case navigations, we allow arbitrary multiplicities since, in general, an event may belong to multiple cases. Consider, e.g., the economic transaction of a purchase between two persons, in a situation where the person is marked as case class. In this setting, the transaction may be considered as an event belonging to the trace of the buyer *and* to that of the seller.

Event-timestamp navigations are of a different nature: since each event must be unambiguously associated to a single timestamp, the navigation can only traverse *functional* associations, that is, many-to-one or one-to-one associations, and lead to a functional attribute.

In both situations, we allow for optionality, i.e., for navigations traversing associations whose minimum multiplicity is 0. This is needed to reflect that a trace may be incomplete (thus missing events), and that multiple traces of the same case class may indeed contain different events. The following example clarifies this aspect.

Example 2. Consider again the class diagram in Fig. 2. The `Invoice` class may be annotated with two event types, one for the invoice registration, and one for its payment. The first event exists for every invoice, as it is associated to the mandatory `invCrTime` timestamp attribute. The second event, instead, does not necessarily exist for every invoice: there may be invoices that are not yet paid, and invoices that may never be paid. This is reflected by the fact that the `payTime` attribute of `Invoice` is optional. ■

In general, there exist several possible event-case and event-timestamp navigational paths. To disambiguate which ones are actually used to define an event class, attribute annotations are used.

Attribute annotations. Attribute annotations decorate event annotations with information about their features. Each attribute annotation consists of an attribute and of the specification of a navigational path to (functionally) reach its value. Mandatory attributes are: *(i) case* (how to reach the case class from the event class); *(ii) timestamp* (how to reach the timestamp attribute for the event class); and *(iii) activity* (a constant string or a navigational path specifying which is the activity to which the event refers). Optional attributes are related to *resources*, i.e., how to reach the identifier and/or the role of the resource responsible for the event.

Example 3. Consider again the App case study. Differently from the manual approach illustrated in Sect. 2.2, in our methodology the four event types elicited by EBITmax in Sect. 2.1 may be elicited as shown in the orange UML notes of Fig. 2:

– Each instance of `ActivePO` (directly corresponding to a case) may determine a SubmitOrder event that occurs at the submission time (attribute `subTime`) for that order.
– Each instance of `TD` may determine a GetTD event for the order obtained by navigating the `refers_to` association, and that occurs at the registration time (attribute `regTime`) for that document.
– Each instance of `Invoice` may determine two events for the order obtained by navigating the `is_for` association: a RegisterInvoice event occurring at the creation time (attribute `invCrTime`) for that invoice, and a PaySupplier event occurring at the payment time (attribute `payTime`) for that invoice. ■

3.3 Automated Event Log Extraction

Thanks to the technique introduced in [8], once the conceptual data model is suitably annotated, it is possible to automatically extract from the legacy information system an event log that reorganizes the data contained there according to the specified annotations. Intuitively, this is done by combining the mapping specification with the annotations, and by computing the answers to a series of queries that ask for all the cases present in the information system and, for each case, all the events referring to that case, together with the corresponding attribute values. The so-obtained data structure can then be represented using the IEEE XES standard for event logs.

Differently from the manual extraction methodology of Sect. 2.2, this approach does not only help the process mining expert in working at the conceptual level without coding ad-hoc views, but also facilitates multi-perspective process mining, that is, the extraction of several event logs reflecting different perspectives over the same information system. Each perspective, in fact, requires to change the annotations, while the conceptual data model $\mathcal{T}$ and the mapping specification $\mathcal{M}$ from the information system to the data model remain unchanged.

4 Implementing the Framework: The onprom Tool

To support the various phases of the OBDA-based event log extraction framework illustrated in Sect. 3, we have developed a tool suite named onprom, consisting of various plug-ins of the ProM extensible process mining framework[14] [4]. Specifically, onprom consists of the following components: *(i)* a *UML Editor*, used to design the domain ontology (cf. Sect. 3.1); *(ii)* an *Annotation Editor*, allowing domain expert to specify the event data annotations (cf. Sect. 3.2); *(iii)* a *Log Extractor*, used to extract from the underlying database the XES event log, based on the annotated domain ontology and the mapping specification (cf. Sect. 3.3). The different tools are implemented as separate projects in Java. When used as ProM plug-ins, they exchange data relying on the mechanisms built in ProM. However, both the UML Editor and the Annotation Editor can be used also as stand-alone tools that operate using files for input and output. We now describe the tools in more detail, relying on the App case study for examples.

4.1 UML Editor

The log extraction framework based on OBDA makes use of a domain ontology expressed in the OWL2 QL profile [12] of OWL2, which is the profile supported by the OBDA system ontop. Ontologies expressed in OWL2 QL admit a natural graphical representation in the form of UML class diagrams [10]. Hence, to provide domain experts support for the design of such ontologies, we have developed a graphical editor for UML class diagrams. Actually, since the editor can import standard OWL2 QL ontologies, it can also be used to modify and enhance an already existing or independently developed OWL2 QL ontology.

To maintain the UML Editor lightweight, and to guarantee at the same time that the designed UML class diagrams can indeed be expressed in OWL2 QL, we have made some natural simplifying assumptions on the form of the UML class diagrams supported by the tool:

– we do not support *completeness* of UML generalization hierarchies, since the presence of such construct would fundamentally undermine the virtual OBDA approach based on query reformulation [10];

[14] http://www.promtools.org/.

– in line with Semantic Web languages, we support explicitly only associations of arity 2, and do not support association classes currently;
– multiplicities in associations (resp., of attributes) are restricted to be either 0 or 1. Hence, we can express functionality and mandatory participation;
– we do not support ISA between associations;
– we ignore all those features of UML class diagrams that are more relevant for the software engineering perspective, and less for the conceptual perspective of UML, such as stereotypes, method specifications, and aggregations.

The developed UML class diagram can be saved in a proprietary JSON format for further processing and as input for the Annotation Editor. It can also be exported as a standard OWL2 QL ontology, hence ready to be processed by ontop. We observe that the graphical layout information, which is not part of the OWL2 QL language, is maintained in the form of OWL2 annotations, thus resulting in an ontology fully compliant with the W3C standard.

A screenshot of the UML Editor with the domain ontology of the App use case is shown in Fig. 5.

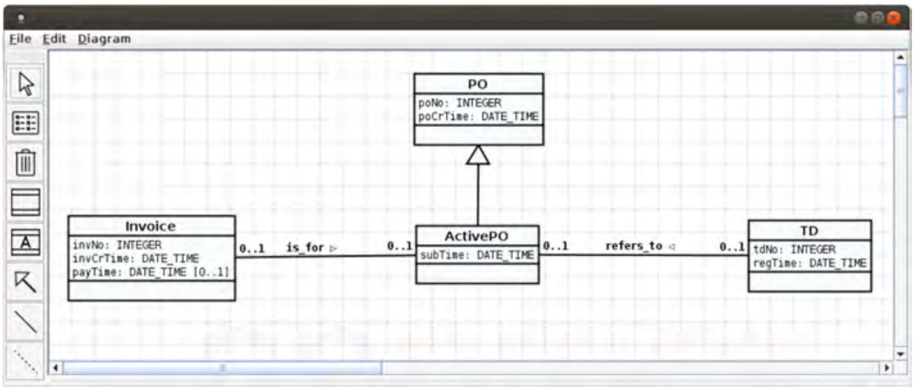


Fig. 5. The UML Editor showing the domain ontology of the App use case

4.2 Annotation Editor

To provide process mining experts with the possibility of specifying in a simple, intuitive way the over the domain-specific ontology $\mathcal{T}$, we have developed an Annotation Editor that supports the different forms of annotation.

A screenshot of the Annotation Editor, with the domain ontology of the App use case annotated with the case and various event elements, is shown in Fig. 6. Specifically, the domain expert has annotated *PO* as the case of the log, while four different events are defined:

1. GetTD accesses the case using the `refers_to` association and navigating the *IS-A* relationship; it has `regTime` in the TD class as timestamp (see Fig. 7a).

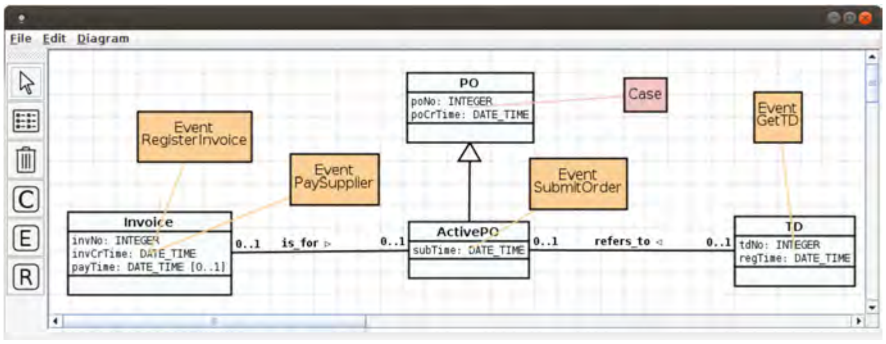
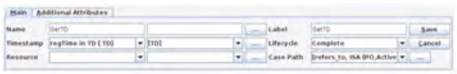
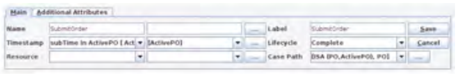


Fig. 6. The Annotation Editor showing annotations for the APP use case



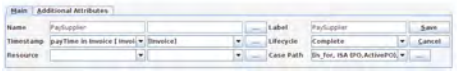
Name: GetTD
Timestamp: regTime in TD [TD] | [TD]
Resource:
Label: GetTD
Lifecycle: Complete
Case Path: [refers_to, ISA [PO,Active

(a) The GetTD event



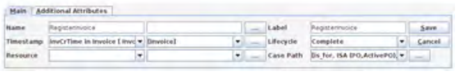
Name: SubmitOrder
Timestamp: subTime in ActivePO [Act | [ActivePO]
Resource:
Label: SubmitOrder
Lifecycle: Complete
Case Path: [ISA [PO,ActivePO], PO]

(b) The SubmitOrder event



Name: PaySupplier
Timestamp: payTime in Invoice [Invoi | [Invoice]
Resource:
Label: PaySupplier
Lifecycle: Complete
Case Path: [is_for, ISA [PO,ActivePO],

(c) The PaySupplier event



Name: RegisterInvoice
Timestamp: invCrTime in Invoice [Invc | [Invoice]
Resource:
Label: RegisterInvoice
Lifecycle: Complete
Case Path: [is_for, ISA [PO,ActivePO],

(d) The RegisterInvoice event

Fig. 7. The properties of event annotations defined for the APP use case

2. SubmitOrder accesses the case directly through an *ISA* relationship; it has `subTime` in the `ActivePO` class as timestamp (see Fig. 7b)
3. PaySupplier accesses the case using the `is_for` association and *IS-A*; it has `payTime` in the `Invoice` class as timestamp (see Fig. 7c).
4. RegisterInvoice also accesses the case using the `is_for` association and *IS-A*, but it has `invCrTime` in the `Invoice` class as timestamp (see Fig. 7d).

We observe that all events have a *complete* life-cycle.

To simplify the annotation task, the editor supports some advanced operations:

– Properties and paths can be chosen using navigational selection over the diagram via mouse-click operations.
– The editor takes into account multiplicities on associations and attributes; when the user is selecting properties of the case and of events (in particular the timestamp), the editor enables only navigation paths that are functional, thus guaranteeing that the properties to include in the extracted log are uniquely determined.

The annotated domain ontology can be exported using a proprietary JSON format, that can then be imported by the log extraction plug-in.

4.3 Log Extraction Plug-in

The two previous plug-ins support the design phase of the log extraction framework (cf. Sect. 3.2). The last plug-in is deployed in the event log extraction phase (cf. Sect. 3.3) to support the automated extraction of event logs that are compatible with XES. The plug-in makes use of the following inputs:

- the information system $\mathcal{I} = \langle \mathcal{R}, \mathcal{D} \rangle$, with the corresponding database schema $\mathcal{R}$;
- the domain ontology $\mathcal{T}$, e.g., as generated via the UML Editor;
- the mapping specification $\mathcal{M}$ between $\mathcal{T}$ and $\mathcal{R}$. Currently, we assume that $\mathcal{M}$ is derived (either semi-automatically or manually) using third-party tools, such as the ontop mapping editor;
- annotations $\mathcal{L}$, which are created using the Annotation Editor.

The tool exploits the *query rewriting* functionalities provided by ontop to generate from the above inputs a *new mapping specification* $\mathcal{M}_{log}$, which establishes a direct correspondence between the data $\mathcal{D}$ in $\mathcal{I}$ and the elements of XES, i.e., trace, event, ..., essentially bypassing the domain ontology $\mathcal{T}$. An ontology capturing the main elements of XES, together with $\mathcal{M}_{log}$ and $\mathcal{I}$, constitutes a new OBDA system that is then used for the event log extraction, by relying on the *data access* functionalities of ontop.

5 Conclusion

In this work, we have presented the onprom framework for the extraction of event logs from legacy information systems, using a real case study to illustrate the limitations of manual extraction, and to show the features of our approach. The framework comes with a methodology centered around conceptual models to capture domain knowledge, to link such knowledge to the underlying data, and to annotate such knowledge with event-related information, reflecting the chosen perspective for process mining. In addition, the framework comes with a toolchain to handle such conceptual models, and automatically extract a XES event log in accordance with the chosen perspective, leveraging ontology-based data access (OBDA) techniques. The toolchain exploits the OBDA features offered by the ontop system, and is fully integrated with the ProM process mining framework. It can be downloaded from http://onprom.inf.unibz.it.

We are currently investigating, together with the EBITmax company, the concrete application of onprom to the Markas case study reported here, and plan to use the results of this case study as a way to further validate the methodology, and to conduct an extensive experimental evaluation, extending the preliminary results obtained in [8].

In addition, we are actively working on extending the annotation editor, on the one hand to provide more guidance to the user in discovering meaningful event classes, and on the other hand to support more sophisticated navigational queries. Finally, we observe that, currently, we do not offer an editor for the specification of mappings that is fully integrated with our toolchain, but we rely instead to what is natively offered by the ontop OBDA framework. A natural next step is to manage the specification of mappings within our toolchain, leveraging recent approaches on the graphical specification of mappings, developed within the recently concluded Optique EU Project[15].

Acknowledgement. This research has been partially supported by the Euregio IPN12 *"KAOS: Knowledge-Aware Operational Support"* project, which is funded by the "European Region Tyrol-South Tyrol-Trentino" (EGTC) under the first call for basic research projects and by the UNIBZ internal project *"OnProm"*. We thank Ario Santoso for the development of the log extraction plug-in of onprom, and Wil van der Aalst for the interesting discussions and insights on the problem of extracting event logs from legacy information systems.

References

1. van der Aalst, W.M.P., et al.: Process mining manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM 2011. LNBIP, vol. 99, pp. 169–194. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28108-2_19
2. van der Aalst, W.M.P.: Process Mining - Data Science in Action, 2nd edn. Springer, Heidelberg (2016)
3. IEEE Computational Intelligence Society: IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams. IEEE Std 1849-2016, i–50 (2016)
4. Verbeek, H.M.W., Buijs, J.C.A.M., Dongen, B.F., van der Aalst, W.M.P.: XES, XESame, and ProM 6. In: Soffer, P., Proper, E. (eds.) CAiSE Forum 2010. LNBIP, vol. 72, pp. 60–75. Springer, Heidelberg (2011). doi:10.1007/978-3-642-17722-4_5
5. Günther, C.W., van der Aalst, W.M.P.: A generic import framework for process event logs. In: Eder, J., Dustdar, S. (eds.) BPM 2006. LNCS, vol. 4103, pp. 81–92. Springer, Heidelberg (2006). doi:10.1007/11837862_10
6. van der Aalst, W.M.P.: Extracting event data from databases to unleash process mining. In: vom Brocke, J., Schmiedel, T. (eds.) BPM - Driving Innovation in a Digital World. Management for Professionals, pp. 105–128. Springer, Cham (2015). doi:10.1007/978-3-319-14430-6_8
7. Syamsiyah, A., van Dongen, B.F., van der Aalst, W.M.P.: DB-XES: enabling process discovery in the large. In: Proceedings of the 6th International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA). CEUR, vol. 1757, pp. 63–77. ceur-ws.org (2016)
8. Calvanese, D., Montali, M., Syamsiyah, A., van der Aalst, W.M.P.: Ontology-driven extraction of event logs from relational databases. In: Reichert, M., Reijers, H.A. (eds.) BPM 2015. LNBIP, vol. 256, pp. 140–153. Springer, Cham (2016). doi:10.1007/978-3-319-42887-1_12
9. Poggi, A., Lembo, D., Calvanese, D., Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. In: Spaccapietra, S. (ed.) Journal on Data Semantics X. LNCS, vol. 4900, pp. 133–173. Springer, Heidelberg (2008). doi:10.1007/978-3-540-77688-8_5

[15] http://optique-project.eu.

10. Calvanese, D., Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R.: Ontologies and databases: the *DL-Lite* approach. In: Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M.-C., Schmidt, R.A. (eds.) Reasoning Web 2009. LNCS, vol. 5689, pp. 255–356. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03754-2_7
11. Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., Xiao, G.: Ontop: answering SPARQL queries over relational databases. Semant. Web J. **8**(3), 471–487 (2017). doi:10.3233/SW-160217
12. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language profiles, 2nd edn. W3C Recommendation, W3C, December 2012. http://www.w3.org/TR/owl2-profiles/
13. Antonioli, N., Castanò, F., Coletta, S., Grossi, S., Lembo, D., Lenzerini, M., Poggi, A., Virardi, E., Castracane, P.: Ontology-based data management for the Italian public debt. In: Proceedings of FOIS. Frontiers in Artificial Intelligence and Applications, vol. 267, pp. 372–385. IOS Press (2014)
14. Jiménez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I., Pinkel, C., Skjæveland, M.G., Thorstensen, E., Mora, J.: BootOX: bootstrapping OWL 2 ontologies and R2RML mappings from relational databases. In: Proceedings of ISWC Posters & Demonstrations Track. CEUR, vol. 1486. ceur-ws.org (2015)