Semantic-Based Process Analysis[*]

Chiara Di Francescomarino[1], Francesco Corcoglioniti[1], Mauro Dragoni[1],
Piergiorgio Bertoli[2], Roberto Tiella[1], Chiara Ghidini[1], Michele Nori[2],
and Marco Pistore[2]

[1] FBK—IRST, Trento, Italy
{dfmchiara,corcoglio,dragoni,tiella,ghidini}@fbk.eu
[2] SayService, Trento, Italy
{bertoli,nori,pistore}@sayservice.it

Abstract. The widespread adoption of Information Technology systems and their capability to trace data about process executions has made available Information Technology data for the analysis of process executions. Meanwhile, at business level, static and procedural knowledge, which can be exploited to analyze and reason on data, is often available. In this paper we aim at providing an approach that, combining static and procedural aspects, business and data levels and exploiting semantic-based techniques allows business analysts to infer knowledge and use it to analyze system executions. The proposed solution has been implemented using current scalable Semantic Web technologies, that offer the possibility to keep the advantages of semantic-based reasoning with non-trivial quantities of data.

1 Introduction

The last decades have witnessed a rapid and widespread adoption of Information Technology (IT) to support business activities in all phases, governing the execution of business processes and the processing and storage of related documents. This, together with knowledge at the business level, gives the potential to leverage IT techniques to analyze business procedures, thus bringing several remarkable advantages, as for example to allow business analysts to observe and analyze process executions; to identify bottlenecks and opportunities of improvement of processes; to identify discrepancies between the way processes have been designed, and the way they are really executed.

In fact, a variety of Business Intelligence tools have been proposed, even by major vendors, that aim at supporting Business Activity Monitoring (BAM), and hence the activities above, to different extent; examples are Engineering's eBAM,[1] Microsoft's BAM suite in BizTalk,[2] Oracle's BAM.[3] However, all these approaches mainly focus, besides data, on the only procedural knowledge. The knowledge related to the static

[*] This work is supported by "ProMo - A Collaborative Agile Approach to Model and Monitor Service-Based Business Processes", funded by the Operational Programme "Fondo Europeo di Sviluppo Regionale (FESR) 2007-2013 of the Province of Trento, Italy.

[1] http://www.eclipse.org/ebam

[2] https://www.microsoft.com/biztalk/en/us/business-activity-monitoring.aspx

[3] http://www.oracle.com/technetwork/middleware/bam/overview/index.html

P. Mika et al. (Eds.) ISWC 2014, Part II, LNCS 8797, pp. 228–243, 2014.

aspects of the domain (e.g., concerning documental or organizational aspects), generally representable as domain ontologies, is not taken into account, thus precluding the capability to reason on and analyze execution data from a business domain perspective.

On the contrary, existing approaches for the semantic monitoring and analysis of processes usually separately focus on model or execution aspects. Several works, including [1,2,3,4,5,6,7], enrich process models with semantic knowledge either for specifying the execution semantics of models or for denoting the meaning of their elements. Only few attempts have been made to combine the static and the procedural model at the business layer with execution data (e.g., [8]). In these works, however, business knowledge is mainly exploited to provide domain-independent solutions to automate the reconciliation between the business and data layers, rather than to support analysis. In this paper we focus on enabling business analysts to perform useful analysis on process execution data, covering both static and procedural dimensions as well as business and data levels. Our contribution is twofold: (i) by extending our previous work [6,9] combining static and procedural dimensions, we define a semantic model for combining static, procedural and data knowledge, which enables semantic reasoning and allows analysts to query asserted and inferred knowledge to bring execution data analysis at business level; and (ii) we propose an implementation of the approach on top of current Semantic Web technologies – namely triplestores – that aims at coping with the large quantities of data and the high data rates typical of real application scenarios.

The paper is organized as follows. In Section 2 we present the problem through an example scenario, which is then used, in Section 3, to introduce the orthogonal dimensions we tackle and the high level idea of the proposed approach. In Section 4 we describe the integrated model and its components from a conceptual point of view, while Section 5 and Section 6 report about implementation and evaluation on a project use case, respectively. Finally, Section 7 presents related works and Section 8 concludes.

2 Scenario

In this section we present an application scenario for the proposed approach. It will be used throughout the paper to clarify the concepts and the motivation behind the work. The example has been taken from an industrial case study related to the Public Administration field in the context of the *ProMo* project for the collaborative modeling, monitoring and analysis of business processes. Figure 1 shows the process model (represented in the BPMN [10] notation) describing the Italian birth management procedure, that aims at recording citizens' birth data. Data have to be stored both in the municipality registry and in the central national registry (SAIA) – since newborns are Italian citizens who live in a given municipality – as well as in the national welfare system repository (APSS) – as newborns are users of the welfare system. For instance, the newborn can be first registered in the welfare repository and then in the registry systems or viceversa, according to the parents' choice. This, in turn, requires the coordination of several actors, as well as the generation of some crucial data that will become part of the personal data of the newborn – e.g. his/her ID, the Fiscal Code (FC).

In scenarios like this one, which are applied on a large scale and with high public costs, it is important to be able to analyze the actual executions of the procedures or to

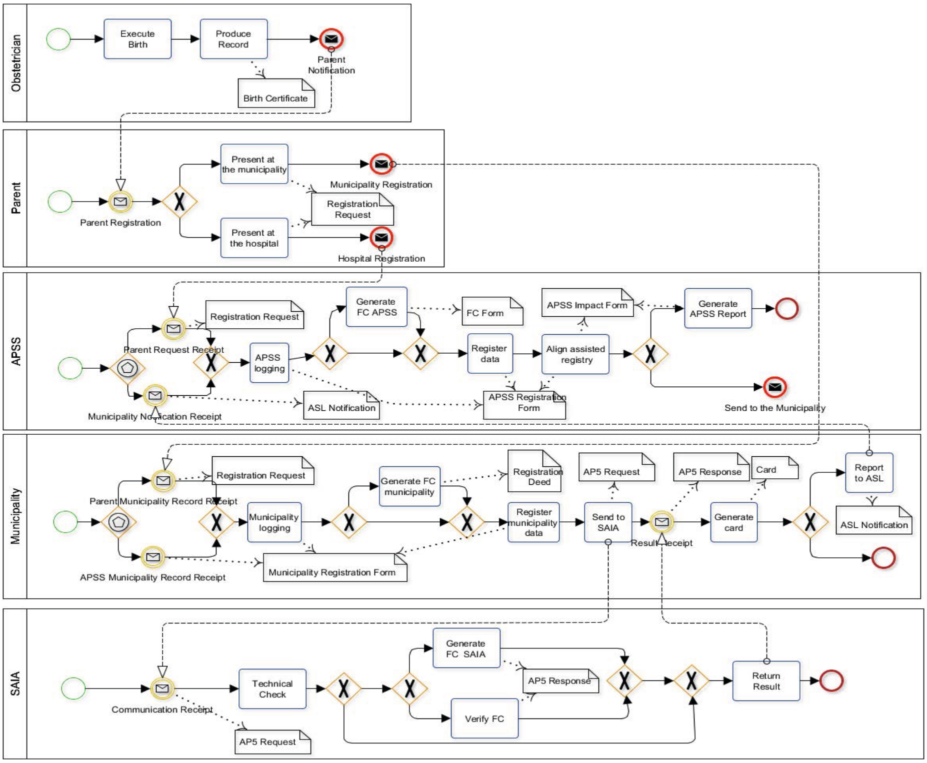

Fig. 1. Birth Management Process

realize what-if analyses for maintenance purposes. This means understanding how procedures perform; how many, when and where failures and bottlenecks occur; whether executions deviate from the model. Examples of queries related to the birth management procedure that business analysts could be interested to answer are:

Q.1 the average time per process execution spent by the municipality of Trento;
Q.2 the total number of Registration Request documents filled from January, 1st, 2014;
Q.3 the percentage of times in which the flow followed is the one which passes first through the APSS pool and then through the Municipality one;
Q.4 the number of cases and the average time spent by each public office involved in the birth management procedure for executing optional activities (i.e., activities which, taken a path on the model, can be either executed or not);

Finding an answer to this kind of questions poses three interesting challenges.

Challenge 1. Combining Three Different Dimensions. Scenarios in which a procedure is carried out in a specific domain, as the one described above, are very common in practice. They are usually characterized by three main dimensions that need to be taken into account per se and in combination in order to be able to analyse the scenario: a *procedural dimension*, a dimension describing the specific *domain of interest* and an *execution dimension*. The *procedural dimension* is defined by the process model describing the carried out procedure. For example, such a dimension is required for detecting in Q.3

whether the execution flow passes first through the APSS pool and then through the municipality pool or viceversa. Knowledge about the *domain of interest*, i.e., the domain in which the procedure is carried out, makes it possible, for example, to identify in Q.2 what are Registration Request documents. Finally, the *execution dimension*, i.e., the actual execution data both in terms of the procedural execution trace and of produced data is required for example for retrieving in Q.1 the actual execution time.

Challenge 2. Semantic Reasoning. In many cases and especially in complex scenarios, data explicitly asserted in collected execution traces and in process and domain models represent only part of the information that is globally available about a process. In these cases, semantic reasoning enabled on top of the three orthogonal components makes it possible to query not only asserted but also inferrable information. For example, in query Q.4 semantic reasoning allows both making explicit the semantics of public offices, as well as reasoning about paths in the process model to detect optional activities (which are situated on alternative paths between two directly connected gateways).

Challenge 3. Scalability. Scenarios characterizing complex organizations, as, for example, Public Administrations (PAs), tend to deal with massive data. PAs usually have huge and complicated procedures with several variants and exceptions, which relate to as much huge and structured domains (e.g., in all PA domains the document classification is usually very intricate). In this kind of scenarios a huge quantity of data is usually produced at a very high rate. For example, in Italy there are about 500000 newborns per year with, on average, a birth per minute.[4] It means that, at least 500000 (one per minute) execution traces of the birth management procedure are produced per year and have to be readily managed and analyzed. This demands for a scalable system able to manage this huge quantity of data in a reasonable time, so as to (i) process and store execution data with a high throughput; and (ii) provide a prompt answer, which also takes into account procedures not yet completed, to queries involving this data.

3 An Integrated View

Analysts dealing with situations like those characterizing complex organizations usually need to face the problem of combining and reconciling knowledge and information related to different orthogonal dimensions, the business and the data as well as the static and the dynamic one. Figure 2 depicts these layers and dimensions:

- At **business level**, knowledge describes the domain of interest (e.g., the company's documents, organization and procedures) and pertains to two dimensions:
  D.1 the *procedural* knowledge (**P**), usually represented using business process models that offer a view on the steps carried out to realize specific objectives;
  D.2 the (static) *domain* knowledge (**K**), which describes aspects of the domain such as the organization structure (e.g., role hierarchy), the data structures (e.g., document organization) and the relations among these and other domain entities; these aspects are usually described in terms of ontological knowledge.
- At **data level**, the data stored by information systems provide information on the actual *executions* of business procedures, leading to our third dimension:

[4] Data by the Italian statistical office for year 2011 (`http://www.istat.it/it/archivio/74300`).

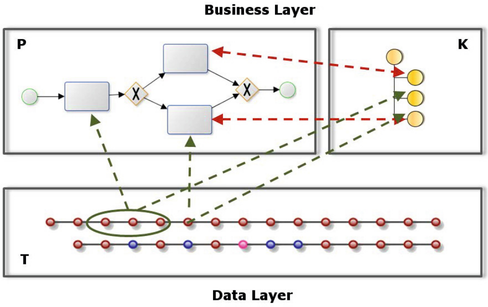

Fig. 2. An integrated view

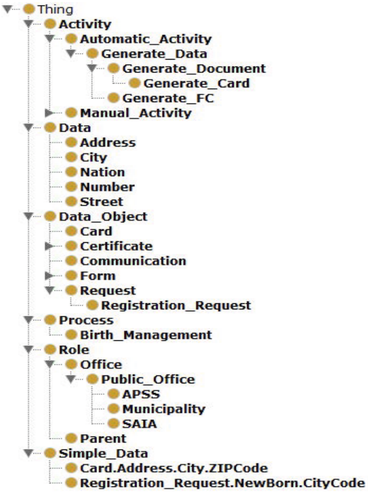

Fig. 3. Extract of domain ontology

D.3 the IT data, i.e., the execution traces (**T**) collected by IT systems that describe the sequence of events and the data related to a process execution.

Concerning IT data, we focus on knowledge integration and exploitation assuming that heterogeneous data have already been reconciled and collected into execution traces. A preprocessing step to collect IT data into execution traces (e.g., [11]) and to align traces with the business layer (e.g., semantic-based [8] or structural approaches [12]) can be realized by exploiting existing approaches in the literature.

In order to improve on state-of-the-art approaches, which do not allow analysts to investigate these three dimensions together, we propose to combine the knowledge related to the domain described at business level K, the one related to the procedural aspects P and the data captured by IT systems T, into a semantic solution and take advantage of it for inferencing and querying purposes.

The first step towards the integration is combining the two types of business knowledge: the static (K) and the procedural (P) one. Intuitively, combining K and P, means enriching the process diagram components with the semantic knowledge [6,9] and viceversa, i.e., adding a procedural view to the static knowledge. For instance, in the example of the birth management procedure, a role `APSS`, semantically characterized as a `Public Office` role by the taxonomy of roles shown in the excerpt of domain ontology of Figure 3, can be associated to the APSS pool of the process in Figure 1.

The second step consists of combining the business and the data layers, i.e., K and P with T. Assuming to have (i) an execution trace that collects all the events and the associated data and performers related to a single execution , and (ii) an alignment of these components with the business knowledge, combining the business and the data layers intuitively means instantiating the business level components with the corresponding set of execution data. For example, an event “Fiscal Code Generation” in an execution trace, according to the alignment between data and business, could be an instance of the activity labeled with “Generate FC Municipality” enriched with its domain semantics.

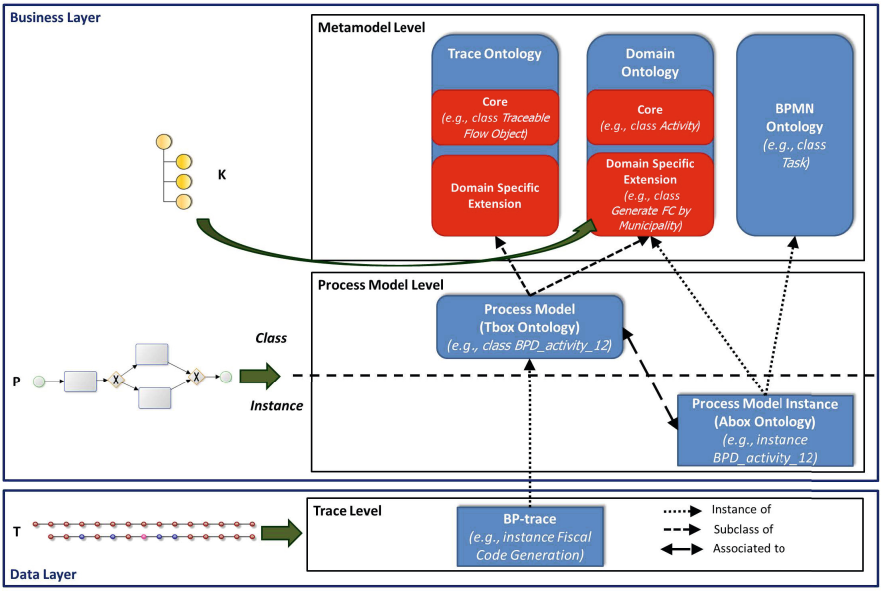

Fig. 4. Integrated Ontological Model. For each component of the model, a class or instance example is provided (among parenthesis).

4 A Comprehensive Model for Process Execution Analysis

To be exploited for query purposes, the three types of knowledge (K, P and T) described in Section 2 need to be expressed in a unique language and plugged into an integrated system. In this section we present the conceptual model used for the integration. To this purpose, three ontologies were formalized – a *BPMN Ontology*, a *Domain Ontology* and a *Trace Ontology* – and a three-level *architectural model* was built on top of them. Purpose of the *Domain Ontology* is to provide a formal description of the static aspects of the domain (K), while the *BPMN Ontology* and the *Trace Ontology* provide a formalization of a metamodel of the procedural (P) and data (T) aspects, respectively.

Figure 4 shows how the three ontologies are used to formalize and combine the three starting ingredients (P, K and T) along with the three levels. Specifically, at the *Metamodel* level, which deals with all types of knowledge encompassing process model scenarios (i.e., the core parts of the *Trace Ontology*, *Domain Ontology* and *BPMN Ontology*), K is formalized as an extension of the core *Domain Ontology*. At the *Process Model* level, which deals with knowledge specific to a process model, the integration of P and K is formalized. Finally, at the *Trace* level, focusing on the execution traces, T and its relationships with P and K are formalized.

In the following we discuss in more details each of the three ontologies (Section 4.1) and then describe their integration into the three-level architectural model (Section 4.2).

4.1 Component Description

BPMN Ontology. The *BPMN Ontology* (BPMNO) formalizes the structure of a business process model diagram (BPD). It is a formalization of the BPMN standard as

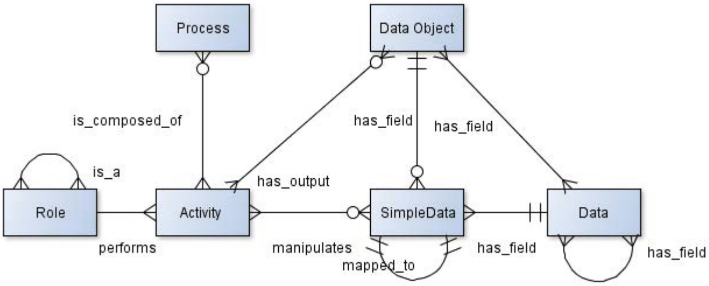

Fig. 5. *Core Domain Ontology*

described in Annex B of the BPMN specification 1.0 [13] enriched with the main components of BPMN2.0 [10], and consists of a set of axioms that describe the BPMN elements and the way in which they can be combined for the construction of BPDs.[5] A detailed description is provided in [6,9]. We remark that the BPMNO provides a formalization of the structural part of BPDs, describing which are the basic elements of a BPD and how they are (can be) connected. The BPMNO is not intended to model the dynamic behavior of BPDs (that is, how the flow proceeds within a process).

Domain Ontology. The *Domain Ontology*, which is in charge of providing a formalization of the static aspects of the domain of interest, is composed of an upper-level part – the *Core Domain Ontology* – that describes the high level concepts and is independent from the specific domain, and, for each domain, of a domain-dependent extension.

Goal of the *Core Domain Ontology* (reported in Figure 5) is offering a framework for the definition of the key entities and relationships characterizing the domain of a business process. Its central entity is the *activity*, which can constitute the building block of a *process*, i.e., a process *is composed of* a non-empty set of activities. An activity is usually performed by an executor with a specific *role*, which, in turn, can be classified in a hierarchy of roles (e.g., in the case of complex organizations) and can produce in output *data objects* (e.g., a document), which are collections of *data*. Data in data objects are organized in data structures, which, in turn, can be further structured in other data structures and so on, so that, only at the end of the chain, a value can be associated to a *simple data* (e.g., a non-decomposable field of a document).

The *domain-dependent extension* specializes classes and properties of the *Core Domain Ontology* for a specific domain of interest, as shown in Figure 3 for the birth management scenario. In this example, the activity of type `Generate_card` performed by the `Municipality` role could have as output a `Card` data object. The `Address` is one of the data reported in the data object. However, `Address` could be, in turn, a data structure organized in `Street`, `Number`, `City`, `Nation`. Also `City` can be further structured (e.g., in terms of `CityName` and `ZIPCode`) and so on up to reach simple data.

It is worth pointing out that the *Core Domain Ontology* defines and emphasizes the relationship between activities and simple data, i.e., how an activity *manipulates* (cre-

[5] The BPMN 1.0 ontology enriched with some of the new main elements of BPMN2.0 can be found at `https://shell-static.fbk.eu/resources/ontologies/bpmn2_ontology.owl`

ates, updates, displays) a simple data. To this end, a unique semantics is associated to each simple data nested in a specific chain of data structures and on each manipulation property defined between manipulating activities and data fields. For example, `Card.Address.City.ZIPCode` is a simple data on which it is possible to explicitly define the activity manipulations. Moreover, simple data of different data objects can be mapped one to another (via property *mapped_to*, see Figure 5) to indicate that they hold the same value, thus providing useful knowledge to perform reasoning. For example, the data on the card emitted by the municipality corresponds to the data reported by the citizen on the registration request module, thus `RegistrationRequest.NewBorn.CityCode` and `Card.Address.City.ZIPCode` are mapped one to another.

Trace Ontology. The *Trace Ontology* ontology has finally been designed to specifically address the aspects related to the execution, providing a metamodel representation of the knowledge that can be collected and traced by IT systems (i.e., knowledge in terms of instances) and stored in execution traces. It also has a core part – the *Core Trace Ontology* – whose key entity is the *Trace*. A trace is composed of *Traceable Process Elements*, which can be either *Traceable Flow Objects* and *Traceable Data Objects*. The first are (instances of) events traced by IT systems that can have both a start and an end time intervals and an actual *Performer*. The latter are (instances of) data structures collecting sets of data. The core part of the *Trace Ontology* (the *Core Trace Ontology*) can then be enriched and structured according to the specific types of information collected by IT systems and analyzed in a particular scenario. For example, properties defining the provenance of data or events of the execution trace (e.g., the IT system they come from), could need to be stored, and hence the *Trace Ontology* extended accordingly.

4.2 A Three-Level Architectural Model

Starting from the components previously described, a three-level architectural model has been built to combine the three dimensions K, P and T. Figure 4 depicts such a model, which extends our previous work [6,9] that only accounts for K and P.

The first level (*Metamodel* level) contains the three ontologies mentioned in the previous subsection, i.e., the initial building blocks of the integrated model: the *BPMN Ontology*, the *Domain Ontology*, representing an ontological formalization of the domain knowledge, and the *Trace Ontology*. Referring to the examples shown (within parenthesis) in Figure 4, classes `Task`, `Activity` and `Traceable_Flow_Object` are representative of the contents of the *BPMN Ontology*, *Domain Ontology* and *Trace Ontology* respectively, and all of them lie in the Metamodel level.

The second level (*Process Model* level) contains an integrated description of a single process model diagram in terms of domain and procedural aspects. Such an integrated model can be looked in two different ways. On one side (and similarly to the approach in [6,9]), a process diagram and the elements it contains are *instances* of the corresponding semantic classes in the *Domain Ontology* and *BPMN Ontology*. On the other side (and differently from [6,9]), a process model diagram and its elements are model components which, though still inheriting their domain semantics from a *Domain Ontology*, act as *classes* that are instantiated by process execution traces. In this view, a more agile and specific semantics than the heavy one given by the BPMN notation, can be provided by these classes to the execution traces (e.g., there is no need to constraint the event corresponding to an activity execution to have at most an outgoing sequence flow). For example, in the birth management process, the BPMN activity labeled with "Generate FC Municipality" (e.g., the diagram element with id "BPD_activity_12") is an instance of the *Domain Ontology* class `Generate_FC_by_Municipality` and of the *BPMN Ontology* class `Task` (see the examples in Figure 4). However, the same element is also a class instantiated by the actual executions (in the trace level) of the "Generate FC Municipality" activity, that, by exploiting subsumption relations, inherits the characteristics of `Traceable_ Flow_Object` from the *Trace Ontology* and of `Generate_FC_by_Municipality` in the *Domain Ontology*. To represent these two perspectives on process model elements, we decided to model them both as individuals of an *Abox Ontology* (as modelling is at instance level) that instantiates the *Domain Ontology* and the *BPMN Ontology*, and as classes of a *Tbox Ontology* (as modelling is at the terminological level) that specializes the *Domain Ontology* and the *Trace Ontology*; a special *associated_to* relation links corresponding elements in the two ontologies.[6]

Finally, a third level (*Trace* level) is devoted to store the execution traces. An execution trace is actually an instance of the process model diagram class of the *Tbox Ontology* that is, as described above, a subclass of the *Domain Ontology* class specifying the process domain semantics and of the `Trace` class of the *Trace Ontology*. For example, an execution trace of the birth management process will be an instance of the *Tbox Ontology* class inheriting from the *Domain Ontology* `Birth_Management` class and the *Trace Ontology* `Trace` class, the latter related to properties typical of execution traces. Events and data structures in the execution trace are managed similarly.

By looking at the examples in Figure 4 we can get an overall clarifying view of the three levels and their relationships. For instance, the trace event "Fiscal Code Generation" is represented as an instance of the *Tbox Ontology* class corresponding to the BPMN diagram element "BPD_activity_12". This class, which extends the *Trace Ontology* class `Traceable_Flow_Object` and the *Domain Ontology* class `Generate_FC_by_Municipality`, is *associated_to* the corresponding instance ("BPD_activity_12") in the *Abox Ontology*. The latter, in turn, is an instance of classes `Task` of the *BPMN Ontology* and `Generate_FC_by_Municipality` of the *Domain Ontology*.

5 Architectural Solution

In this section we propose an architecture for the runtime collection of information at the various dimensions (P, K, T), its integration according to the comprehensive model of Section 4 and its unified querying to support business analysts needs.

The two major challenges at the architectural level are to cope with the huge quantity of collected trace data and their fast rate of arrival on the one hand, and to allow analysts to query also for implicit knowledge in collected data on the other hand, which requires some kind of reasoning support in the system (respectively, challenges 3 and 2 of Section 2). To address these challenges, we investigate the use of Semantic Web technologies in the form of *triplestores*. Triplestores are repositories for RDF data, possibly

[6] Despite the described scenario and the technological solution adopted would allow the use of punning, i.e., treating classes as instances of meta-classes, we chose to have separate entities for classes and instances, in order to keep the solution in line with the traditional conceptual distinction between classes and instances.

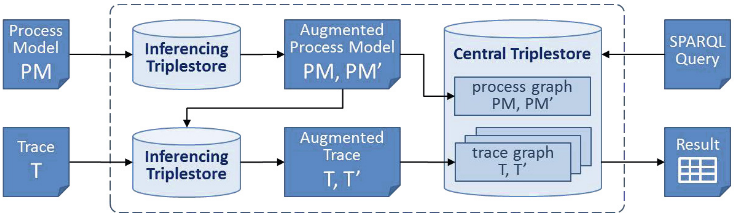

Fig. 6. Architecture. PM and T denote asserted triples, PM' and T' inferred triples.

organized into *named graphs*, with support for reasoning with different semantics (e.g., RDFS, OWL 2 RL and OWL 2 QL) and querying via the standard SPARQL query language and protocol [14]. Triplestore solutions have been widely investigated in recent years as a means to manage huge quantities of data, and several triplestore implementations have become mature products (e.g., OWLIM[7] and Virtuoso[8], to name a few), so they represent a natural choice for implementing our model.

In the rest of the section we detail, referring to Figure 6 for a general overview, how data is organized, populated and queried in our architecture using triplestores, while in the next section we present some results pointing out the potential of the solution.

5.1 Data Organization

The most efficient way to support SPARQL querying is to place all data in a *central triplestore*, materializing all the inferrable triples so that no reasoning has to be done at query time. We thus exclude federated queries over multiple triplestores as well as backward-chaining reasoning approaches (e.g., based on query rewriting), as they both introduce additional delay that is unacceptable for complex analytical queries.

The central triplestore has to store the process model data (the *Tbox Ontology* and *Abox Ontology* of Figure 4, which in turn encompasses both the procedural knowledge P and the static knowledge K) and all the completed/running execution traces collected so far (the IT data T). To better organize this information we store it in different named graphs: one for the process model and one for each trace (see Figure 6). This solution allows for the use of SPARQL constructs related to named graphs (FROM and USING clauses and graph management operations) to more easily select and manipulate traces.

5.2 Data Population

Process model data is produced at design time and can be stored once per all, while a *trace update* operation must occur every time a new piece of information about a running process is captured. Each trace update operation requires either the creation or the modification of the named graph of the trace. While these modifications are often monotonic (i.e., only new data is added), for the sake of generality we consider them as non-monotonic. In particular, non-monotonicity may arise when only a subset of process activities are observed and recorded in a trace (e.g., only e-mail exchanges are

[7] `http://www.ontotext.com/owlim`

[8] `http://virtuoso.openlinksw.com/`

recorded among human activities). In these cases, missing information could be automatically reconstructed in an approximate way based on known data. This could imply the possibility of a revisitation of reconstructed information as new data is observed.

An obvious solution to populate the central triplestore would be to directly access and modify it with collected data, relying on its reasoning capabilities for inferring implicit triples. This approach is however inefficient, as it overloads a single system with querying and reasoning tasks, the latter being particularly expensive due to the need to retract inferred triples that are no longer valid after a non-monotonic update. A better solution can be devised by considering that traces are largely independent one to each other, as they describe different and unrelated instances of documents, process activities and other execution events. This means that no meaningful knowledge can be inferred by combining two different traces,[9] and thus inference over traces can be computed by processing each trace in isolation (together with the process model) and merging all the inferred triples that have been separately computed.

The trace independence assumption leads naturally to the processing scheme shown in Figure 6. When the system is first started, the process model (PM) is read and the inference machinery of a temporary *inferencing triplestore* is employed to augment it with inferred triples (PM'), producing an *augmented process model* that is stored once per all in the *central triplestore*. Whenever a trace update operation is triggered, the trace data (T) is fed into another temporary *inferencing triplestore* together with the TBox definitions of the augmented process model, producing an *augmented trace* containing inferred triples (T'). The augmented trace is then extracted from the inferencing triplestore, filtered to remove triples of the augmented process model (as already stored) and triples that are not needed by queries[10] (for efficiency reasons), and finally stored in a trace-specific named graph in the central triplestore.

A first benefit of the described scheme is a clear separation of reasoning and querying through the use of separate triplestores, which can be chosen and optimized based on each particular task. To that respect, in our implementation we use OWLIM-Lite for both tasks, configuring OWL 2 RL inference and no persistence when used as an inferencing triplestore, and disabling inference and increasing index sizes when using it as the central triplestore. A second and more important benefit, however, is the possibility to parallelize trace update operations using multiple worker threads and/or machines, thus enabling massive scalability. The only 'bottleneck' is represented by the storage of processed data in the central triplestore, but this operation is very efficient as it does not involve any reasoning or inference retraction.

5.3 Data Querying

As shown in Figure 6, SPARQL queries by business analysts are targeted at the central repository, where they are evaluated against the integrated knowledge base built and

[9] We refer here to inference at the ABox level (the trace data) based on the OWL 2 semantics. The limited overlapping in terms of instances (e.g., documents) among traces means that little or nothing about an instance in a trace can be inferred in OWL 2 exploiting knowledge about unrelated instances in other traces. This does not exclude, however, the possibility to 'infer' useful knowledge by comparing or aggregating trace data in a non-OWL setting, a task supported by the querying facilities of our approach.

[10] In detail, we drop unnecessary `x owl:sameAs x` and `x rdf:type owl_restriction_bnode` triples.

augmented with inferred triples as previously described. As an example, Listing 1 reports the formulation in SPARQL of query Q.4, whose results are shown in Table 1. In general, we found analytical queries to greatly benefit from SPARQL 1.1 aggregates, but SPARQL support for managing dates and other temporal information (e.g., for computing time differences) resulted quite inadequate. We addressed this problem by defining a suite of user-defined SPARQL functions that can be used by analysts, leveraging the extension mechanisms provided by Sesame,[11] though this solution is generally not portable across different triplestore implementations.

```
PREFIX bpmn:    <http://dkm.fbk.eu/index.php/BPMN_Ontology#>
PREFIX domain:  <https://dkm.fbk.eu/#>
PREFIX trace:   <http://dkm.fbk.eu/Trace_Ontology#>
PREFIX ttrace:  <http://dkm.fbk.eu/TTrace_Ontology#>
PREFIX fn:      <http://shell.fbk.eu/custom_functions/>

SELECT ?office_name ?activity_name (COUNT(?t) AS ?executions) (AVG(?t) AS ?time)
WHERE {
  [] a bpmn:business_process_diagram , domain:Birth_Management; # retrieve process
    bpmn:has_business_process_diagram_pools [                   # and public office
      bpmn:has_pool_participant_ref [                           # participants for
        a domain:Public_Office;                                 # each pool of the
        bpmn:has_participant_name ?office_name ];               # birth management
      bpmn:has_pool_process_ref ?process ].                     # process diagram

  ?gateway1 a bpmn:gateway.                                     # require the
  ?gateway2 a bpmn:gateway.                                     # presence of two
  ?activity a bpmn:activity.                                    # gateways and
                                                                # an activity in
  ?process bpmn:has_process_graphical_elements                  # the participant
      ?gateway1 , ?gateway2 , ?activity.                        # process

  ?gateway1                                                     # require two paths
    bpmn:is_directly_connected_via_sequence_flow ?gateway2;     # between gateways:
    bpmn:is_connected_via_sequence_flow ?activity.              # - a direct path
                                                                # - an indirect
  ?activity ttrace:associated_to ?Activity ;                    #   path passing
    bpmn:has_flow_object_name ?activity_name;                   #   through the
    bpmn:is_connected_via_sequence_flow ?gateway2.              #   activity

  OPTIONAL {                                                    # if the activity
    ?activity_execution a ?Activity;                            # was performed,
    trace:initial_start_dateTime ?start;                        # get its execution
    trace:final_end_dateTime ?end.                              # time using the
    BIND ((fn:timestamp(?end) - fn:timestamp(?start)) AS ?t)    # custom function
  }                                                             # fn:timestamp
}
GROUP BY ?office_name ?activity_name
```

Listing 1. SPARQL formulation of query Q.4: "Number of cases and avg. time spent for optional activities by public offices involved in the birth management procedure".

Table 1. Query results

office_name	activity_name	executions	time
"SAIA"	"Verifica CF"	86	501.097 s
"SAIA"	"Genera CF SAIA"	15009	270.122 s
"Comune"	"Genera CF Comune"	21000	485.541 s
"APSS"	"Genera CF APSS"	8315	418.327 s

[11] http://www.openrdf.org/

6 Evaluation

In this section we report on the evaluation of the proposed model and architecture on a real case study – the birth management scenario –investigated in the scope of the *ProMo* project, with the goal to assess the usefulness and scalability of the approach.

In the above mentioned scenario:

- P is a BPMN process model (see Figure 1) containing 4 pools, 19 activities, 11 domain objects, 19 events, 14 gateways, 54 sequence flows and 6 message flows;
- K is a domain ontology (an extract is shown in Figure 3) containing 5 properties and 379 classes covering 28 activities and 12 data objects such that, on average, each data object contains 25 simple data fields organized in a 4 levels-depth structure;
- T is a set of execution traces automatically generated based on the aforementioned P and K and a few samples of real traces (that we can not directly use as containing sensitive personal data); on average each trace covers 10 events with associated data objects, and is encoded with 2040 triples from which 1260 triples can be inferred.

In order to assess the approach in real contexts, we selected and analyzed 8 queries among those that business analysts involved in the *ProMo* project were interested to investigate, with the selection driven by our perception about their importance for business analysts as well as by the goal to cover as much variety as possible (e.g., type of query result, multiple VS single trace analysis) so to increase their representativeness. Queries and their analysis are reported in Table 2. For each query, the columns corresponding to the component(s) (P, K or T) involved by the query, are marked. Similarly, the inference column (Inf.) is marked when the query demands for reasoning support. The table suggests that all the three dimensions explored by the proposed approach, as well as the inference support, are indeed needed to answer business analysts' queries, thus suggesting the usefulness of the proposed approach in real business scenarios.

Concerning scalability, we mainly focus on the storing and querying aspects. Assuming the data rates of the Italian birth management scenario at national level (Section 2), we investigated a one day, one week and one month loads (respectively ∼1500, ∼10500 and ∼42000 traces). Table 3 shows the corresponding performance figures, measured on a quad-core Intel Core I7 860 workstation with 16 GB of memory. For each load, we

Table 2. Query analysis

Query	Description	P	K	T	Inf.
Q.1	Average time per process execution spent by the municipality of Trento			X	
Q.2	Total number of Registration Request documents filled from Jan 1st, 2014		X	X	
Q.3	Percentage of times in which the flow followed is the one which passes first through the APSS pool and then through the municipality one	X		X	
Q.4	Number of cases and average time spent by each public office involved in the birth management procedure for executing optional activities	X	X	X	X
Q.5	Number of times the municipality sends to SAIA a request without FC	X	X	X	X
Q.6	Last event of trace TRACEID			X	
Q.7	Average time spent by trace TRACEID			X	
Q.8	Does trace TRACEID go through activity labeled "Present at the hospital"?	X		X	

Table 3. Scalability Results

Traces	Stored triples			Storing		Querying	
	Asserted	Inferred	Total	Throughput	Total time	Time Q.4	Time Q.8
1500	3062349	1895471	4957820	37.89 trace/min	2426.88 s	324 ms	41.4 ms
10500	21910269	13057464	34967773	37.41 trace/min	16851.21 s	881.4 ms	26.2 ms
42000	87503538	52045200	139548738	37.34 trace/min	67537.95 s	4510.0 ms	105.0 ms

report the number of stored triples (asserted, inferred, total), the storing time (throughput in traces per minute and total population time) and the average evaluation times for queries Q.4 and Q.8 (from Table 2), which are representative, respectively, of analytical and non-selective queries and of very specific and selective queries.

Overall, the system is able to manage a throughput of about 37 traces per minute, which is perfectly adequate with the Italian scenario (a newbirth per minute). More significantly, the throughput is largely independent of the load, demonstrating how the choice to decouple inference for each trace allows to efficiently cope with increasingly large amounts of data. Finally, the time required for performing a query as complex as Q.4, which involves all the dimensions and exploits inferred knowledge (e.g., for identifying public offices and determining whether an activity is optional), is still acceptable for the specific context and for an online analysis of data. Of course, the more the repository grows, the slower the answer is. Nevertheless, queries concerning the whole set of data collected in a month, can be managed in times of the order of seconds.

7 Related Works

Approaches adding formal semantics to process models (also defined in the BPMN notation) are not new in the literature [1,2,3,4,5,6]. We can divide the existing proposals into two groups: (1) those adding semantics to specify the dynamic behavior exhibited by a business process [1,2,3], and (2) those adding semantics to specify the meaning of the entities of a Business Process Diagram (BPD) in order to improve the automation of business process management [4,5,6]. The latter are more relevant in our context, as they focus on models, rather than the execution semantics of the processes.

Thomas and Fellmann [5] consider the problem of augmenting EPC process models with semantic annotations. They propose a framework which joins process model and ontology through properties (such as the “semantic type” of a process element). This enrichment associates annotation properties to the process instances. In the SUPER project [4], the SUPER ontology is used for the creation of semantic annotations of both BPMN and EPC process models in order to support automated composition, mediation and execution. In [1], semantic annotations are introduced for validation purposes, i.e. to verify constraints about the process execution semantics.

In our previous work [6,9], we enrich process models with semantic knowledge and establish a set of subsumption (aka subclass) relations between the classes of two ontologies: one formalizing the notation used (the BPMN meta-model) and another describing the specific domain of the process model. This way we provide an ontology integration scheme, based on hierarchical ontology merge, that supports automated verification of semantic constraints defining the correctness of semantic process annotations as well as of structural constraints [6].

Only few works (e.g., [15,8,16]) in the context of the SUPER project have tried to combine static and procedural business level aspects (very close to the ones we consider) with the execution data. Nevertheless either they try to provide a comprehensive Semantic Business Process Management framework (e.g., [15]) or, when explicitly dealing with process monitoring and analysis [8], they mainly focus on the usage of semantic technologies to provide domain-independent solutions to process monitoring, which, instantiated by concrete organizations in their specific domain, aims at automate the reconciliation between the business and the data level. In this work, semantic technologies are exploited to integrate the three investigated dimensions, thus supporting process analysis by means of inference and querying.

Finally, there exist works dealing with the formalization of abstract frameworks for describing process executions. For instance, the standardized PROV-O [17] ontology, whose aim is providing a formalization for representing and interchanging provenance information generated in different systems, provides a set of classes (and relationships and axioms) to be instantiated with activity executions and artefacts, similarly to the *Trace Ontology*. Through the opportune mapping between PROV-O and *Trace Ontology* classes (e.g., the PROV-O `activity` and the *Trace Ontology* `Traceable_Flow_Object`), it would be possible to align the proposed model to PROV-O and thus allow the consumption of *Trace Ontology* data by PROV-O aware applications.

The problem of exploiting the advantages deriving from semantic-based approaches with large amounts of data has been widely investigated in the last years. One of the most known and used solutions is the Ontology Based Data Access [18] (OBDA). OBDA aims at providing access to data stored in relational heterogeneous data sources through the mediation of a semantic layer in the form of an ontology, that provides a high level conceptual view of the domain of interest [19]. Among the available efficient query answering reasoners we can find MASTRO-i [18] and the most recent Quest [20].

Triplestore solutions like Virtuoso, Owlim, Bigdata and others provide similar query answering functionalities though a different mechanism. Among the available Semantic Web technologies we chose to use a semantic triplestore solution. Indeed, besides allowing us to address the semantic as well as the big data requirements, it also allows us to update the model structure in a lightweight way. Being based on triples, it does not strongly constraint the meta-level, thus leaving the flexibility to change the model in an agile way, without loosing the data already stored.

8 Conclusion

In this paper we show how to combine different orthogonal dimensions and exploit reasoning services in order to expose interesting analysis on organizations' execution data in business terms. In detail, static domain knowledge, procedural domain knowledge and execution data have been plugged into a semantic solution and Semantic Web technologies have been exploited to cope with large quantities of data. The approach has been applied to an industrial case study investigated in the context of the *ProMo* project for the modeling, monitoring and analysis of Italian Public Administration procedures. In the future we plan to further investigate the reasoning capabilities that semantic technologies can offer, also in terms of user-defined rule-sets.

References

1. Weber, I., Hoffmann, J., Mendling, J.: Semantic business process validation. In: Proc. of Semantic Business Process and Product Lifecycle Management workshop (SBPM) (2008)
2. Wong, P.Y., Gibbons, J.: A relative timed semantics for BPMN. Electronic Notes in Theoretical Computer Science 229(2), 59–75 (2009); Proc. of 7th Int. Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA 2008)
3. Koschmider, A., Oberweis, A.: Ontology based business process description. In: Proceedings of the CAiSE 2005 Workshops. LNCS, pp. 321–333. Springer (2005)
4. Dimitrov, M., Simov, A., Stein, S., Konstantinov, M.: A BPMO based semantic business process modelling environment. In: Proc. of SBPLM at ESWC. CEUR-WS (2007)
5. Thomas, O., Fellmann, M.: Semantic EPC: Enhancing process modeling using ontology languages. In: Proc. of Semantic Business Process and Product Lifecycle Management workshop (SBPM), pp. 64–75 (2007)
6. Di Francescomarino, C., Ghidini, C., Rospocher, M., Serafini, L., Tonella, P.: Reasoning on semantically annotated processes. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 132–146. Springer, Heidelberg (2008)
7. Tomasi, A., Marchetto, A., Di Francescomarino, C., Susi, A.: reBPMN: Recovering and reducing business processes. In: ICSM, pp. 666–669 (2012)
8. Pedrinaci, C., Lambert, D., Wetzstein, B., van Lessen, T., Cekov, L., Dimitrov, M.: SENTINEL: A semantic business process monitoring tool. In: Proc. of Ontology-supported Business Intelligence (OBI), pp. 1:1–1:12. ACM (2008)
9. Di Francescomarino, C., Ghidini, C., Rospocher, M., Serafini, L., Tonella, P.: Semantically-aided business process modeling. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 114–129. Springer, Heidelberg (2009)
10. Object Management Group (OMG): Business process model and notation (BPMN) version 2.0. Standard (2011)
11. Bertoli, P., Kazhamiakin, R., Nori, M., Pistore, M.: SMART: Modeling and monitoring support for business process coordination in dynamic environments. In: Abramowicz, W., Domingue, J., Węcel, K. (eds.) BIS Workshops 2012. LNBIP, vol. 127, pp. 243–254. Springer, Heidelberg (2012)
12. de Leoni, M., Maggi, F.M., van der Aalst, W.M.P.: Aligning event logs and declarative process models for conformance checking. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 82–97. Springer, Heidelberg (2012)
13. Business Process Management Initiative (BPMI): Business process modeling notation: Specification (2006), `http://www.bpmn.org`
14. Seaborne, A., Harris, S.: SPARQL 1.1 Query Language. Recommendation, W3C (2013)
15. Hepp, M., Roman, D.: An ontology framework for semantic business process management. In: Wirtschaftsinformatik (1), Universitaetsverlag Karlsruhe, pp. 423–440 (2007)
16. Pedrinaci, C., Domingue, J., Alves de Medeiros, A.K.: A core ontology for business process analysis. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 49–64. Springer, Heidelberg (2008)
17. Sahoo, S., Lebo, T., McGuinness, D.: PROV-O: The PROV ontology. Recommendation, W3C (2013)
18. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The MASTRO system for ontology-based data access. Semant. Web 2(1), 43–53 (2011)
19. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Conceptual modeling for data integration. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Mylopoulos Festschrift. LNCS, vol. 5600, pp. 173–197. Springer, Heidelberg (2009)
20. Rodriguez-Muro, M., Calvanese, D.: Quest, an OWL 2 QL reasoner for ontology-based data access. In: OWLED, vol. 849. CEUR-WS.org (2012)