Semantic Access to Siemens Streaming Data: the OPTIQUE Way*

E. Kharlamov[1] S. Brandt[2] M. Giese[3] E. Jiménez-Ruiz[1] S. Lamparter[2]
C. Neuenstadt[4] Ö. Özçep[4] C. Pinkel[5] A. Soylu[3,6] D. Zheleznyakov[1]
M. Roshchin[2] S. Watson[7] I. Horrocks[1]

[1] University of Oxford; [2] Siemens CT; [3] University of Oslo; [4] University of Lübeck;
[5] fluid Operations AG; [6] Gjøvik University College; [7] Siemens Energy

1 Introduction

Siemens Energy runs several service centres for power plants. The main task of a service centre is remote monitoring and diagnostics of many thousands appliances, such as gas and steam turbines, generators, and compressors installed in plants. Monitoring and diagnostics are performed by service engineers and are typically conducted in four steps: *(i)* engineers receive a notification about a potential or detected issue with an appliance, *(ii)* they gather data relevant to the case, *(iii)* analyse the data, and finally *(iv)* report about ways to address the issue to the appliance owner. Currently, Step (ii) of the process is the bottleneck consuming, as statistics shows [5], up to 80% of the overall time needed by the engineer to accomplish the task. The main reason for this time consumption is the indirect data access, i.e., in many cases the engineers have to access data via IT experts. Involvement of IT experts in the process slows it down dramatically due to various reasons such as their overload, miscommunication between IT experts and the engineers.

Enhancing Step (ii) by enabling direct data access for Siemens' engineers, i.e., without IT experts being involved, is important for enhancing the overall performance of the monitoring [5]. This task is challenging for several reasons. One reason is the conceptual mismatch between the way the engineers expect the data to look like, i.e., the language and structures they use to describe the data, and the way the data is actually described and structured in Siemens databases. Moreover, the data accessible from Siemens service centres have various formats, large in volume, and it changes at a high rate, thus reflecting the dimensions of Big Data. Indeed, it is stored in several thousands databases, its size is in the order of hundreds of terabytes, and it currently grows with the average rate of 30 GB per day. At the moment only Siemens IT experts fully understand these data and thus currently only they can write queries over these databases in order to extract information relevant for engineers.

Ontology Based Data Access (OBDA) [8] has been recently proposed to enhance end-user direct data access. The key idea behind OBDA is to use *ontologies*, to mediate between users and data. Ontologies describe the domain of interest on a higher level of abstraction and in terms that are clear for domain experts. In OBDA users formulate their information needs as queries using terms defined in the ontology, and ontological queries are then translated into SQL or some other database query languages and executed automatically, without an IT expert's intervention. To this end a set of *mappings* describes the relationship between the ontological vocabulary and the data schema.

* This research has been partially supported by the EU project Optique (FP7-IP-318338), the Royal Society, the EPSRC grants Score!, DBonto, and MaSI[3].

```
CREATE STREAM S_out AS
CONSTRUCT    GRAPH NOW { ?c2 rdf:type :MonInc }
FROM         STREAM S_Msmt [NOW-10s, NOW]->"1S"^^xsd:duration,
             STATIC ABOX <http://www.optique-project.eu/siemens/ABoxstatic>,
             TBOX <http://www.optique-project.eu/siemens/TBox>
USING        PULSE WITH START = "00:10:00CET", FREQUENCY = "1S"
WHERE        {?c1 a sie:Assembly. ?c2 a sie:Sensor. ?c1 sie:inAssembly ?c2.}
SEQUENCE BY  StdSeq AS seq
HAVING       MONOTONIC.HAVING(?c2,sie:hasValue)

CREATE AGGREGATE MONOTONIC:HAVING ($var,$att) AS
HAVING EXISTS ?k IN SEQ: GRAPH ?k { ?c2 sie:showsFailure } AND
       FORALL ?i < ?j  IN seq, ?x, ?y:
       IF ( ?i,  ?j < ?k  AND GRAPH ?i {$var $attr ?x} AND GRAPH ?j {$var $attr ?y})
       THEN ?x<=?y
```

Fig. 1. Monitoring task in STARQL, where the prefix *sie* stands for www.siemens.com/demo#

In this demo we present our OBDA solution for Siemens Energy [5] which we develop as a part of the OPTIQUE project [3, 4] The demo will focus on three aspects of the solution: *(i) deployment module*: to semi-automatically deploy the platform over the Siemens relational data streams with the help of ontology and mapping bootstrapping and importing, *(ii) component for formulation and registration of monitoring tasks*: to create general monitoring tasks as parametrised continuous queries and register concrete instances of these tasks over specific data streams, *(iii) monitoring dashboards*: to visualise results of monitoring tasks. We demonstrate OPTIQUE on an anonymized version of Siemens data streams gathered from 200 gas/steam turbines between 2002 and 2011 and using 20 representative monitoring tasks from Siemens service centres that are gathered as a part of requirement analyses for developing our OBDA solution [5].

2 System Overview

In this section we give details of our three OBDA components. All components are integrated in a unified OPTIQUE solution that is based on Information Workbench, a generic and extensible platform for semantic data management that provides many base components for our solution such as APIs, datastores, and visualisation engines.

Deployment Support. In order to deploy an OBDA system over relational data streams, one has to develop a new or reuse a third party domain ontology that describes the domain behind the streams. E.g., for Siemens monitoring the ontology should describe structure and functionality of turbines, technical characteristics of turbines' components including deployed sensors. Besides, the ontology should capture additional relevant information such as results of previous monitoring tasks and repairs, weather forecasts, etc. Creating such ontology and then developing mappings to relate it to Siemens databases is a costly process. We addressed this problem by providing a tool BOOTOX [2, 7] for semi-automatic extraction of OWL 2 ontologies and R2RML mappings from relational databases and for incorporation of third party OWL 2 ontologies in an existing OBDA system deployment either by aligning it to the ontology used in the deployment or by connecting it to the databases underlying the deployment with so called direct mappings. We applied our deployment support solution to Siemens data and obtained an OWL 2 QL ontology with dozens classes and properties, and about 50 axioms. Our OBDA platform also allows for mapping editing and we used the editor to complement the semi-automatic deployment with another dozen of classes and properties, and 20 axioms. You can see a screenshot of our deployment module in Figure 2.

Monitoring Queries. In order to express monitoring tasks we developed *(i)* a devoted query language STARQL [6] for temporal and streaming queries and *(ii)* a visual query

system OPTIQUEVQS for end users without a prior knowledge of formal query languages that allows to formulate conjunctive STARQL queries. The main features of STARQL are: it allows to express typical mathematical, statistical, and event pattern features needed in real-time monitoring scenarios; it comes with a formal syntax and semantics [5, 6] that combines open and closed-world reasoning and extends snapshot semantics for window operators [1] with a sequencing semantics that can handle integrity constraints such as functionality assertions. In spite of its expressivity, answering STARQL queries is still efficient since they can be transformed into relational stream queries. Both inputs and outputs of STARQL queries are timestamped RDF triples. Therefore, triples, coming from the result of one query, can be used as input when constructing another query. While producing a STARQL query, one can select an ontology and streams over which the query will be evaluated.

Consider an example of a monitoring task: *Detect a real-time failure of the turbine caused by the a temperature increase within 10 seconds.* This task can be formulated in STARQL as in Figure 1. An output stream `S_out` is defined by the following language constructs: The `CONSTRUCT` specifies the format of the output stream, here instantiated by RDF triples asserting that there was a monotonic increase. The `FROM` clause specifies the resources on which the query is evaluated: the ontology (here separated in TBox containing intensional knowledge and the ABox containing factual data), and the input stream(s) for which a window operator is specified with window range (here 10 seconds) and with slide (here 1 second). The `PULSE` declaration specifies the output frequency. In the `WHERE` clause bindings for sensors (attached to some assembly) are chosen. For every binding, the relevant condition of the monitoring task is tested on the window contents. Here this condition is abbreviated by `MONINC.HAVING(?c, sie:hasValue)` using a macro that is defined at the bottom of Fig. 1 in an `AGGREGATE` declaration. In words, the conditions asks whether there is some state `?k` in the window s.t. the sensor shows a failure message at `?k` and s.t. for all states before `?k` the attribute value `?attr` (in the example instantiated by `sie:hasValue)` is monotonically increasing.

The example query can also be formulated with the help of STREAMVQS, a variant of OPTIQUEVQS [9], see the screenshot in Figure 2. STREAMVQS allows domain experts to construct queries by combining classes and properties using a box-and-arrow visualisation metaphor. During query construction, the user can choose dynamic classes and properties, i.e. properties and classes whose extensions are time dependent. Moreover, the user may specify additional predefined *patterns* of functions on dynamically updated window contents. The idea is that the power user or the IT expert constructs these patterns for each use case by looking at the special needs and re-occurring calculations as well as sub-queries and hence builds a library of relevant patterns—similar to having a library of mappings, ontologies, and queries from which the user can choose.

Diagnostic Dashboard. In order to address diverse needs of end users in answer visualisation we developed a flexible wiki-based *Diagnostic Dashboard* that can be easily customised by end users themselves. The dashboard allows to visualise query answers, inspect query results, do incremental query refinement, and export of relevant result fragments to external diagnostic tools. Moreover, it allows to perform monitoring of incoming data streams and query answers for continuos queries over these streams. In Figure 2, we present four examples of our visualisation widgets. Depending on the type of data (e.g., time series data, appliance structure), a suitable visualisation paradigm has to be selected (e.g., pivot table, trend diagram, histogram). The diagnostic dash-

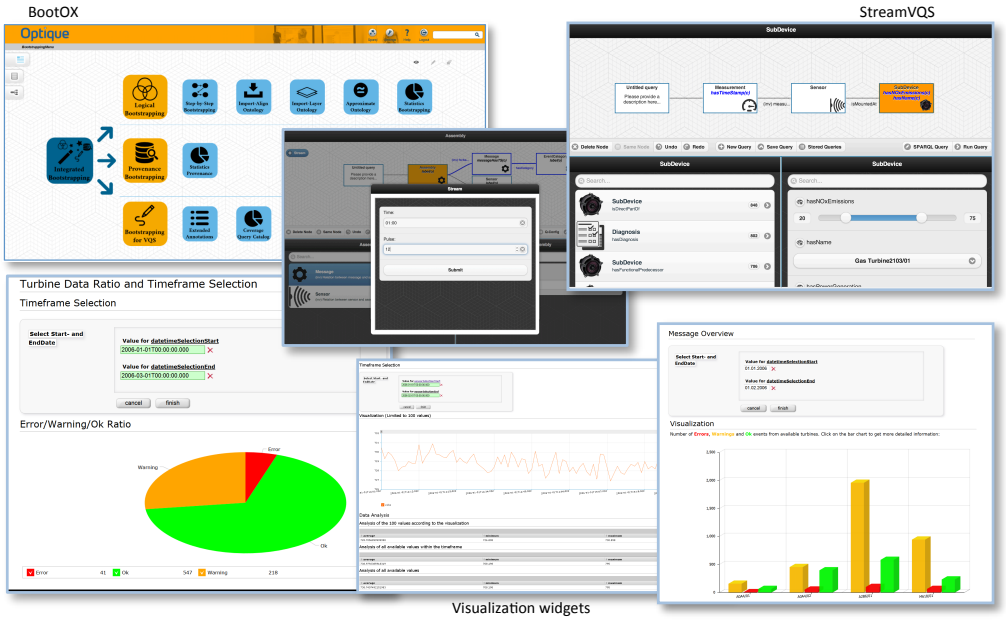

Fig. 2. Screenshots of the OPTIQUE platform deployed over Siemens streaming data

board can also choose the representation paradigm for query answers automatically by analysing the corresponding SPARQL query.

3 Demonstration Scenario

We will demonstrate semantic access to Siemens relational streams with OPTIQUE using anonymised version of streams gathered from 200 Siemens gas and steam turbines between 2002 and 2011, and 20 representative Siemens monitoring tasks. The attendees will be able to deployment of the platform over schemas of the relational streams using bootstrapping and importing. Then, they will be able to register streaming queries by either choosing them from a catalog of preconfigured queries or by formulating them in the query system. Finally, they will be able to run queries and observe the results of the evaluation in the monitoring dashboard.

4 References

[1] A. Arasu et al. The CQL continuous query language: semantic foundations and query execution. In: *The VLDB Journal* 15 (2 2006).

[2] E. Jiménez-Ruiz et al. BootOX: Practical Mapping of RDBs to OWL 2. In: *ISWC*. 2015.

[3] E. Kharlamov et al. Optique: Towards OBDA Systems for Industry. In: *ESWC (SE)*. 2013.

[4] E. Kharlamov et al. Enabling Ontology Based Access at an Oil and Gas Company Statoil. In: *ISWC*. 2015.

[5] E. Kharlamov et al. How Semantic Technologies Can Enhance Data Access at Siemens Energy. In: *ISWC*. 2014.

[6] Ö. L. Özçep et al. A Stream-Temporal Query Language for Ontology Based Data Access. In: *KI 2014*. Vol. 8736. 2014.

[7] C. Pinkel et al. RODI: A Benchmark for Automatic Mapping Generation in Relational-to-Ontology Data Integration. In: *ESWC*. 2015.

[8] A. Poggi et al. Linking Data to Ontologies. In: *J. Data Sem.* 10 (2008).

[9] A. Soylu et al. A Preliminary Approach on Ontology-Based Visual Query Formulation for Big Data. In: *MTSR*. 2013.