How Semantic Technologies can Enhance Data Access at Siemens Energy[†]

Evgeny Kharlamov[1‡], Nina Solomakhina[2], Özgür Özçep[3], Dmitriy Zheleznyakov[1], Thomas Hubauer[2], Steffen Lamparter[2], Mikhail Roshchin[2], and Ahmet Soylu[4]

[1] University of Oxford, UK, [2] Siemens Corporate Technology, Germany,
[3] Hamburg University of Technology, Germany, [4] University of Oslo, Norway

Abstract. We present a description and analysis of the data access challenge in the Siemens Energy. We advocate for Ontology Based Data Access (OBDA) as a suitable Semantic Web driven technology to address the challenge. We derive requirements for applying OBDA in Siemens, review existing OBDA systems and discuss their limitations with respect to the Siemens requirements. We then introduce the Optique platform as a suitable OBDA solution for Siemens. Finally, we describe our preliminary installation and evaluation of the platform in Siemens.

1 Introduction

The growth of available information in enterprises requires new efficient methods for data access by domain experts whose ability to analyse data is at the core of making business decisions. Current centralised approaches, where an IT expert translates the requirements of domain experts into Extract-Transform-Load (ETL) processes to integrate the data and to apply predefined analytical reporting tools, are too heavy-weight and inflexible [1]. In order to support interactive data exploration, domain experts therefore want to access and analyse available data sources *directly*, without IT experts being involved.

This direct data access is particularly important for Siemens Energy[1] that 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 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 reasons including the overload of IT experts and miscommunication between them and the engineers.

Enabling direct data access for engineers in Siemens is a challenging task primarily due to the Big Data dimensions as well as the conceptual mismatch between the language and structures that the engineers use to describe the data, and the way the data is actually described and structured in databases. Regarding the Big Data dimensions, the data

[†] The research was supported by the FP7 grant Optique (n. 318338).

[‡] Corresponding author: evgeny.kharlamov@cs.ox.ac.uk

[1] http://www.energy.siemens.com/

accessible from Siemens service centres naturally reflects the variety, volume, and velocity dimensions of Big Data: it is stored in several thousands databases where many have different schemata, its size is in the order of hundreds of terabytes, and it currently grows with the average rate of 30 GB per day. Regarding the conceptual mismatch, it occurs because industrial schemata are often integrated from autonomously evolving databases that have been adapted over years to the purposes of the applications they underlay, and not to the purpose of being intuitive for domain experts. Only IT experts fully understand this evolving structure of databases and thus currently only they can write queries over these databases in order to extract information relevant for engineers.

Ontology Based Data Access (OBDA) [2] has been recently proposed as a means to enhance end-user direct data access. The key idea behind OBDA is to use *ontologies*, i.e., semantically rich conceptual domain models, 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. Ontologies have become a common and successful mechanism to describe application domains in, e.g., biology, medicine, and the (Semantic) Web [3]. This success is partially due to a number of available formal languages for describing ontologies, including the Web Ontology Language (OWL) standardised by W3C. 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 over the data automatically, without an IT expert's intervention. To this end a set of *mappings* is maintained that describes the relationship between the ontological vocabulary and the schema of the data.

The goal of this paper is to argue that OBDA has a good potential in improving direct access by engineers to the data at Siemens Energy. To this end, in Section 2 we analyse reactive and predictive diagnostics at Siemens and derive five Siemens direct data access requirements. In Section 3 we introduce OBDA, show that it conceptually satisfies the Siemens requirements, and argue that existing OBDA systems are not mature enough to fulfill the requirements. In Section 4 we give a brief overview of the Optique platform, a novel OBDA platform developed within the Optique project [4, 5, 6], and focus on our advances in the development of the platform for Siemens. More precisely, we present a diagnostic dashboard that integrates tools for query formulation together with tools for visualisation of query answers, and our advances in processing timestamped static and streaming data. In Section 5 we present our preliminary deployment of the Optique platform over Siemens data and a preliminary user evaluation. In Section 6 we conclude and discuss the lessons we learned as well as future work.

2 Siemens Monitoring and Diagnostic Service

Siemens produces a variety of rotating appliances, including gas and steam turbines, generators, and compressors. These appliances are complex machines and typically used in different critical processes including power generation where each hour of downtime may cost thousands of euros. Thus, these appliances should be under a constant monitoring that requires an in-depth knowledge about their components and setup. Siemens provides such monitoring via service centres and operates over fifty such centres worldwide, where each centre is responsible for several thousands appliances. Typical monitoring tasks of a service centre include *(i) reactive and preventive diagnostics* of turbines which is about offline data analysis applied after a malfunction or an abnormal behaviour such as vibration, temperature or pressure increase, unexpected events, or even

Fig. 1. High-level view on the turbine service process

unexpected shutdowns, of a unit is detected; *(ii) predictive analysis* of turbine conditions which is about real-time data analyses of data streams received from appliances. We now discuss these monitoring tasks in detail and present requirements to enhance these tasks.

2.1 Reactive and Preventive Diagnostics

Reactive diagnostics is usually applied after a malfunction of a unit has occurred, e.g., a turbine abnormal shutdown. Complementarily, the preventive diagnostic task is performed before a malfunction of a unit, when its abnormal behaviour is detected, e.g., high vibration or temperature increase. Both diagnostic tasks are triggered either when a customer sends a service ticket claiming assistance or an automated diagnostic system creates such a ticket. Figure 1 depicts a general process triggered when a service ticket arrives. We now discuss each step of the process in detail.

Arrival of a service ticket. A service ticket typically contains information on when a problem occurred and its frequency. In some cases the ticket isolates the location of the problem in the appliance and its cause, but often it has no or few details.

Example 1. An example of a reactive monitoring request from a customer is:

> *Figure out why the turbine failed to start during the last five hours, with the goal of checking that there will be no fault of the turbine.*

A typical preventive monitoring request could be

> *Will there be a failure of the turbine after the observed temperature increase?* ■

Data acquisition. Service engineers gather relevant data by querying databases that are updated every hour and on demand and contain sensor and event data. In order to support data gathering, Siemens equips service centres with more than 4,000 predefined queries and query patterns of different complexity. Engineers use the queries by setting parameters such as time periods, names of events or sensors, sensor types, etc.

Example 2. Based on the service ticket of Example 1, the engineer formulates the following information need and has to find appropriate queries to cover it:

> *Return the most frequent start failure and warning messages of the gas turbine T01 during the last week. Moreover, find analogous cases of failures for turbines of the same type as T01 in the last three months.* ■

Query results visualisation. Sensor data is visualised with the use of standard diagrams, and event messages are presented as a list, i.e., as an Excel spreadsheet, with timestamps and additional attributes.

Data preprocessing. The queried data is preprocessed using generic procedures such as sensor check (i.e., whether sensor data quality is appropriate), threshold and trend analysis. Independent from the concrete ticket, these preprocessing steps are done manually, e.g., over the visualised Excel spreadsheets, or using specialised analytic tools.

Data analysis. The engineer uses sophisticated diagnostic models and tools for complex analysis, e.g., Principal Component Analysis or other statistical methods, to detect and isolate the given problem based on the preprocessed data. Typically, analysis tasks are executed individually for each ticket. The gathering and analysis steps are often carried out iteratively, i.e., the results from one iteration are used to pose additional queries.

Report preparation. This process terminates when an explanation for the problem in the service ticket is established. In this case the engineer provides the customer with a report aggregating the result of the analysis and describing possible further actions.

2.2 Predictive Analysis

In predictive analysis, in contrast to the diagnostic process described above, appliances are continuously monitored, i.e., without prior service tickets, using online processing of the incoming sensor data. The other process steps of predictive analysis are similar to the ones described in the previous section, but have to be applied online to streaming data with minimal user intervention. The purpose here is to analyse the current condition of an appliance by combining operating information, system data, specifications of concrete product lines, and temporal phases of operating regimes. This information allows to predict whether some parts of an appliance should be repaired soon, assess risks related to the use of this parts, and adjust maintenance intervals for each part by automatically integrating this information into service scheduling, thus, minimizing maintenance cost.

Example 3. For predictive analysis of turbines, the diagnostic engineer may want to be automatically notified when a turbine shows repetitive start failures combined with increased vibration values during its operating time. This can be formulated as follows:

> *Notify me if a turbine that had more than three start failures in the last two weeks additionally shows abnormal vibration values in operative phases.* ■

2.3 Siemens Requirements

The main bottleneck for diagnostics is the data gathering part, which takes up to 80% of the overall diagnostic time. The main reason is that finding the *right* data for analytics is very hard due to limitations of predefined queries, complexity of data, complexity of query formulation, and limitation to explicitly stated information. We now derive concrete requirements that a system for supporting diagnostic process should fulfill.

R1: Integrated Data Access Siemens data over which the queries are formulated naturally reflects the variety, volume, and velocity dimensions of Big Data. The data is stored in so-called data centres, each responsible for several thousand of appliances such as turbines, where a typical turbine has about 2,000 sensors constantly producing measurements. This data can be roughly grouped into three categories: *(i)* sensor and event data from appliances, *(ii)* analytical data obtained as results of monitoring tasks conducted by service centres for the last several years, and *(iii)* miscellaneous data, typically stored in XML, contains technical description of appliances, types of configurations for appliances, indicates in which databases information from sensors is stored, history of whether forecasts, etc. All in all the data is stored in several thousand databases having a large of different schemata. The size of the data in the order of hundreds of terabytes, e.g., there is about 15 GB of data associated to a single turbine, and it currently grows with the average rate of 30 GB per day. At the moment there is no unified access point to the Siemens data and it is required.

R2: Flexible Definition of Queries Existing predefined queries in the Siemens query catalogue, about 4,000 queries, are often not sufficient to cover information needs as they are often either too general, thus yielding an overload of irrelevant information, or too specific, thus not providing enough relevant data. For gathering relevant data, service engineers often have to use several queries and combine their results. When this is not sufficient, existing queries have to be modified or new queries should be created. To this end the engineer contacts an IT expert and this leads to a complex and time-consuming interaction that takes up to weeks. The reason why it takes so long is miscommunication, high workload of IT personnel, complexity of query formulation, and long query execution times. In average up to 35 queries require modification every month, and up to 10% of queries are changed throughout a year. Moreover, several new queries are developed monthly. Therefore, flexible modification and definition of queries is one of the strong requirements for the improvement of the diagnostic process.

R3: Utilising implicit information In databases it is typically assumed that only explicit data matters, i.e., the data which is stored in the system. From a formal perspective, they adopt the so-called *closed-world* semantics, meaning that exactly the information stated is true, and anything not stated is false. While this perspective may be valid in the context of controlled systems, completeness of data is hardly ever the case in practical industry applications such as the ones in Siemens. Here, the fact that we do not have a measurement tuple for a certain time point does not mean there is no measurement. This could be reflected by the so-called *open-world* semantics, that allows to derive *implicit information* from the data stated explicitly, typically using some forms of background knowledge. This implicit information logically follows from what is stated explicitly, and its use can greatly increase the practical benefit of a diagnostic system.

R4: Stream Data Processing Predictive analysis requires the use of both static information from the past and streaming information on the current status of appliances. Access to data from the past allows to detect, for instance, seasonal patterns. Continuous monitoring of the streaming data provides prognoses for key performance indicators and countermeasures before a system shutdown occurs. Currently, service engineers do not have a direct access to streaming data. However, engineers often need to access event and sensor data from several appliances, and as of now it requires downloading streaming data for each related turbine. One of the requirements for the predictive analysis is the possibility to integrate sensor and event streaming data from several turbines and diagnostic centres and provide the use of continuous queries on data streams.

Summing up on the requirements above, Siemens needs a solution that naturally integrates the existing databases, allows for flexible query definition, exploits both explicit and implicit data, and allows for processing (multiple) data streams. The last technical requirement for a desired solution naturally arises from practical considerations:

R5: System Deployment and Maintenance Support The cost of the deployment and maintenance of the proposed solution should not exceed the benefits from the use of the system. In particular, the solution should support semi-automatic deployment over Siemens databases.

In the next section we present an approach that addresses these requirements.

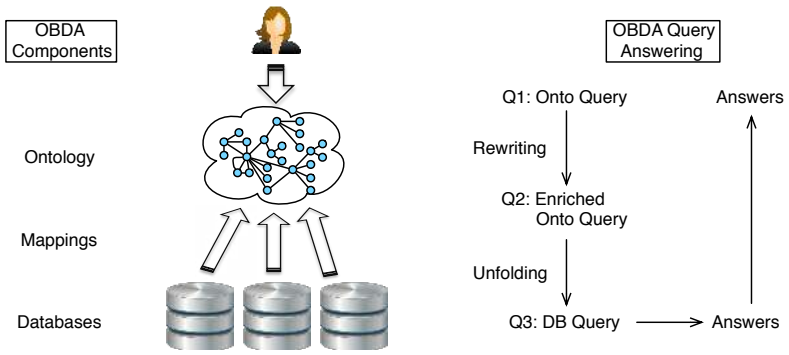

Fig. 2. OBDA: components and a general idea of query processing

3 Ontology Based Data Access

Ontology Based Data Access (OBDA) is a prominent approach for end-user oriented access to databases. OBDA relies on Semantic Web technologies and it has been heavily studied by the Semantic Web community [2, 7].

The main idea behind OBDA is to provide a user with access to the data via an *ontology* that is specific to the user's domain. The ontology can be written in some ontology language, e.g., in the Web Ontology Language OWL 2 standardised by W3C. This ontology hides from the user technical details about the database schemata while it exhibits to the user a domain specific vocabulary of classes and properties i.e., unary and binary predicates, that the user is familiar with. This vocabulary is related to the database schemata via *mappings*, which are declarative specifications, similar to view definitions in databases. There are several mapping languages available, e.g., R2RML standardised by W3C. Figure 2 presents a general conceptual diagram illustrating OBDA: its main components and the workflow of query answering in OBDA systems.

The user formulates queries over ontologies in terms of the classes and properties. The standard query language for ontologies is SPARQL 1.1 standardised by W3C. An ontological query Q_1 is evaluated over databases in three steps. First, Q_1 is expanded with relevant information from the ontology in order to retrieve both explicit and implicit answers from the databases. This is accomplished by *query rewriting*, which takes the query Q_1 and the ontology, and produces the query Q_2. Note that Q_2 is logically equivalent to Q_1 with respect to the ontology while it "absorbs" a fragment of the ontology necessary for retrieving all answers relevant to Q_1. We refer the reader to, e.g., [8], for details on query rewriting techniques. OBDA systems typically do rewriting of so-called *conjunctive queries* with ontologies that fall in the OWL 2 QL profile of OWL 2. This profile is specifically tailored for data access and allows for efficient query processing [8]. As the second step, the query Q_2 is translated using mappings into a query Q_3 over the database schemata, e.g., into SQL when the data is relational. This step is referred to as *unfolding*. Finally, Q_3 is executed over the data by a DBMS and the answers are returned to the user.

We now illustrate OBDA on the following example which is based on the ontology and mappings that we developed for the Siemens use case. Note that, for the sake of clarity, the example is based on simplified versions of these ontology and mappings.

Example 4. The ontology in Figure 3 says that turbines can be either gas or diesel. A gas turbine may have the following parts: *(i)* a control system that in turn has a control

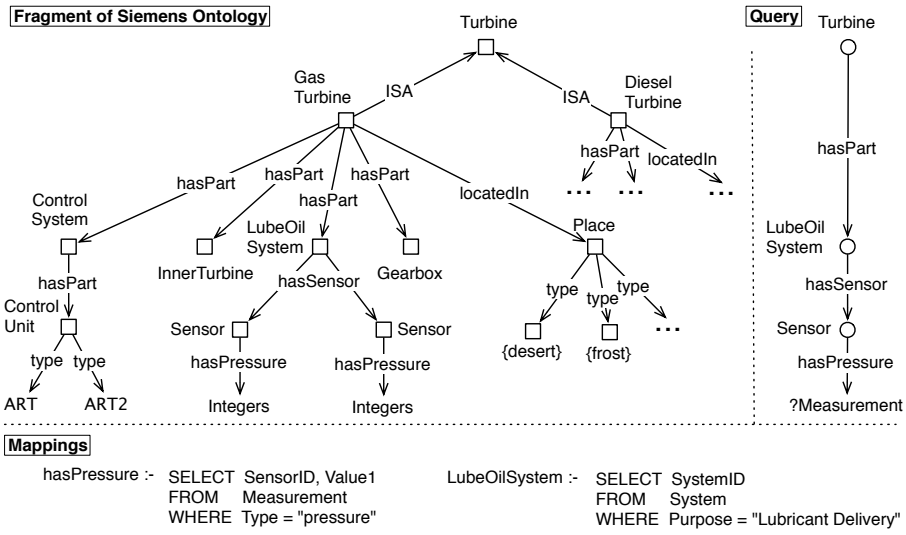

Fig. 3. Simplified Siemens ontology and mappings, example query

unit of types ART or ART2, *(ii)* inner turbine, *(iii)* lube-oil system that may have several sensors for measuring pressure, and *(iv)* gearbox. Moreover, a gas turbine can be located in a place such as a desert, or a frost, etc. For the sake of simplicity, we assume that diesel turbines are modelled in the same way as the gas ones.

The query in Figure 3 asks: "*Return the pressure measured by sensors of lube oil systems in turbines*." This is an ontological query which corresponds to Q_1 in Figure 2. This query can be written in SPARQL as follows:

```
SELECT ?Measurement
WHERE {?X rdf:type siemens:Turbine. ?X siemens:hasPart ?Y.
       ?Y rdf:type siemens:LubeOilSystem.
       ?Y siemens:hasSensor ?Z. ?Z rdf:type siemens:Sensor.
       ?Z siemens:hasPressure ?Measurement.}
```

Query rewriting techniques applied to this query and the turbine ontology produce two more queries that have the same structure, as Q_1, but the first query has Gas Turbine and the second one has Diesel Turbine in the place of Turbine. The query Q_2 is the union of Q_1 with these two queries. In terms of SPARQL, Q_2 can be obtained from Q_1 by substituting the first triple of its WHERE clause with the following expression:

```
{ ?X rdf:type siemens:Turbine } UNION
{ ?X rdf:type siemens:GasTurbine } UNION
{ ?X rdf:type siemens:DieselTurbine }
```

There are two mappings in Figure 3. The left one says how to "populate" the property hasPressure: one has to project tuples of the table Measurement, where the value of the attribute Type is "pressure". The projection on the attribute SensorID gives the subject and on the attribute Value1 gives the object of hasPressure. The right mapping says how to "populate" the class LubeOilSystem: one has to project tuples of the table System where the Purpose is "Lubricant Delivery" on the SystemID attribute. These mappings can be used to unfold the SPARQL query Q_2 into an SQL query Q_3. We do not give Q_3 here due to space limit since this would require to introduce six more mappings. ■

3.1 How OBDA Can Help in Improving Data Access in Siemens

In Section 2.3, we presented five Siemens data access requirements. We will discuss now how OBDA naturally addresses all of them and thus we believe that OBDA has a potential in improving data access in Siemens.

OBDA naturally addresses Requirement R1 on integrated data access since one ontology can mediate the user and several databases with different formats via mappings. Regarding Requirement R2 on flexible definition of queries, since ontologies describe the domain of end users, formulation of queries over ontologies is conceptually much easier than over databases. Thus, by relying on intuitive query formulation tools, users can combine existing queries and write new queries without any knowledge of the schemata of multiple databases residing behind the ontology. Regarding Requirement R3 on utilising implicit information, OBDA naturally does so via logical reasoning during the query rewriting process. Regarding Requirement R4 on stream data processing, OBDA is powerful enough in addressing this, e.g., by employing temporal ontologies, mappings, and queries involving temporal operators. Regarding Requirement R5 on system installation support, there is a body of work on semi-automatic ontology and mapping discovery as well as bootstrapping techniques that allow to extract ontologies and mappings from relational schemata.

Thus, what we need for Siemens is an OBDA system that *(i)* supports distributed data processing, *(ii)* provides a flexible intuitive query formulation and visualisation support, *(iii)* relies on logical reasoning to obtain both explicit and implicit answers, *(iv)* accommodates temporal and data steams, and *(v)* allows to bootstrap, edit, and reuse ontologies and mappings. As we see next, no such OBDA system exists.

3.2 Existing OBDA Systems and Their Limitations

We now show that, despite the recent advances in OBDA systems, they are currently not mature enough to be applied off-the-shelf in Siemens and both theoretical and practical developments are required. There are several academic and industrial systems for OBDA or that are very similar to OBDA in spirit. Mastro [9], morph-RDB [10], and Ontop [11] support ontology reasoning and thus address Requirement R3, while D2RQ [12], OntoQF [13], Virtuoso[2], Spyder[3], and Ultrawrap [14] do not support reasoning and thus fail Requirement R3. Moreover, all these systems fail Requirement R2: Ultrawrap, Ontop, Mastro, and morph-RDB lack user-oriented query formulation interfaces and query visualisation; they provide SPARQL end-points and predefined queries; OntoQF considers ontology queries as OWL statements and has no visual query formulation support. To the best of our knowledge, there are no OBDA systems that support streaming queries of Requirement R4 and address distributed query processing of Requirement R1. Ontop and Mastro support a limited form of data federation, which is not sufficient to allow for data integration of R1. Ontop provides a restricted deployment support in the form of bootstrapping, while OntoQF, Mastro, and morph-RDB lack it. Ultrawrap extended with QODI system [15] allows for so-called query-driven ontology alignment that can allow to import existing ontologies and thus facilitate installation of OBDA systems. Virtuoso, Spyder, and D2RQ provide a limited installation support via direct mapping bootstrappers and simple user interfaces for navigating the data graph. However, no advanced mapping management of Requirement R5 is supported.

[2] http://virtuoso.openlinksw.com/

[3] http://www.revelytix.com/content/spyder

In the following section we will overview the Optique platform [4, 5], that addresses the limitations of existing systems discussed above, and is adopted for Siemens.

4 Enhancing OBDA Technology for Siemens

The Optique platform is an end-to-end OBDA solution, i.e., it supports the whole OBDA cycle from deployment to query answer visualisation. Optique platform integrates a number of existing systems and provides several new components. It was tested with various use cases, including Norwegian Petroleum Directorate Fact Pages [16], and demonstrated in [17].[4] We now give an overview of the platform. Details on the architecture and the individual components of the platform cab be found in [17, 5] and by following the references given below.

4.1 Optique Platform

The system is currently under development and it allows to: *(a)* create and edit mappings, *(b)* create, edit, and import ontologies, *(c)* integrate several relational databases and data streams, *(d)* formulate and visualise one time and continuous queries, and *(e)* browse query results. Thus, the Optique platform satisfies all the Siemens system Requirements R1-R5. Note that the current version provides only a limited support of distributed query processing and a limited treatment of temporal and streaming data.

We now discuss the workflow of the system. For the system deployment, one can bootstrap ontologies and mappings from the underlying relational data sources, incorporate external ontologies into the system, and edit ontologies and mappings [18]. This heavily relies on logical reasoning, for which we use the HermiT ontology reasoner [19]. After the system is deployed, the underlying data sources can be queried via our query formulation tools. They allow to compose queries by navigation over the system's ontology, by writing natural language queries, and rely on ontology reasoning (with HermiT). The formulated queries are internally translated into SPARQL queries and sent to the query transformation engine for processing: rewriting against the ontology and further unfolding into SQL queries with the mappings [20]. For rewriting and unfolding, we rely on the Ontop query transformation engine [11]. SQL queries are executed over the data sources underlying the system by the DBMSs of the sources. For distributed query planning [21] and processing of temporal queries [22], we rely on the ADP system [23], which is a system for large scale elastic data processing on the cloud, which, in particular, is needed to cope with the huge data sets provided by Siemens. Both Ontop and ADP are integral components of the Optique platform. Resulting query answers are visualised using templates and widgets such as tables, timelines, maps, charts, etc., depending on the data modalities. The Optique platform implementation is based on the Information Workbench (IWB) [24], a generic and extensible platform for semantic data management which provides a rich infrastructure for platform.

We now consider two aspects of the platform specific for Siemens in detail: its diagnostics dashboard for query formulation and answer visualisation, and temporal and streaming processing part.

4.2 Diagnostics Dashboard

In order to address diverse needs of end users in query formulation and answer visualisation we developed a flexible wiki-based *Diagnostic Dashboard* that can be easily

[4] Optique demo video: `www.youtube.com/user/optiqueproject/playlists`

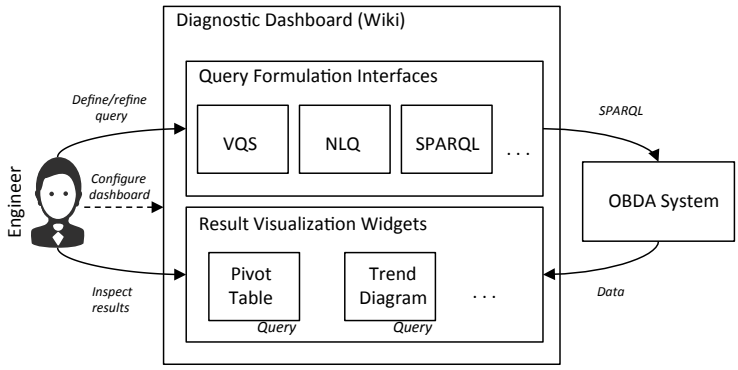

Fig. 4. Components of Diagnostic Dashboard

customised by end users themselves. In Figure 4 we illustrate the engineer's work cycle with the platform through the dashboard. Engineers can define and refine queries via several query formulation tools, that include *(i)* Visual Query System (VQS), *(ii)* Natural Language Query (NLQ) interface, and *(iii)* SPARQL query editor. In the upper part of Figure 5 we present screenshots of our VQS, SPARQL editor embedded in it, as well as our NLQ interface. VQS depicts the domain of interest and queries using a graph based representation paradigm, so end users can directly manipulate visual objects and construct queries. VQL combines query-by-navigation and faceted search techniques over an underlying ontology graph (see [25, 26] for more details). NLQ allows users to specify a query by means of a controlled natural language. Sentences written by engineers are parsed and mapped to concepts, properties, and individuals of the underlying ontology, taking into account ontological axioms. Then, based on additional rules, the the result of the mapping is translated into a SPARQL query (see [27] for more details).

Result visualisation widgets allow to visualise query answers, inspect query results, do incremental query refinement, and export of relevant result fragments to external diagnostic tools. Moreover, the widgets allow to perform monitoring of incoming data streams and query answers for continuos queries over these streams. In the lower part of Figure 5 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 dashboard can also choose the representation paradigm for query answers automatically by analysing the corresponding SPARQL query.

4.3 Temporalised and Streamified OBDA

Recent efforts on temporalised [28, 29] and streamified [30, 31, 32] OBDA systems provide first steps towards handling temporal and streaming data in industrial applications. However, none of these approaches satisfies the requirements of the Siemens use case: either there is no implemented engine or it is still not fully developed (see the benchmark tests in [33]). Besides, in all cases some aspects of query processing in temporalised and streamified OBDA are not addressed, e.g., unfolding with mappings, ontology reasoning.

Streaming and Temporal ontology Access with a Reasoning-based Query Language (*STARQL*) [34, 35] offers a query framework allowing to deal with streams of time-stamped RDF triples on the background of mappings and an ontology. The development

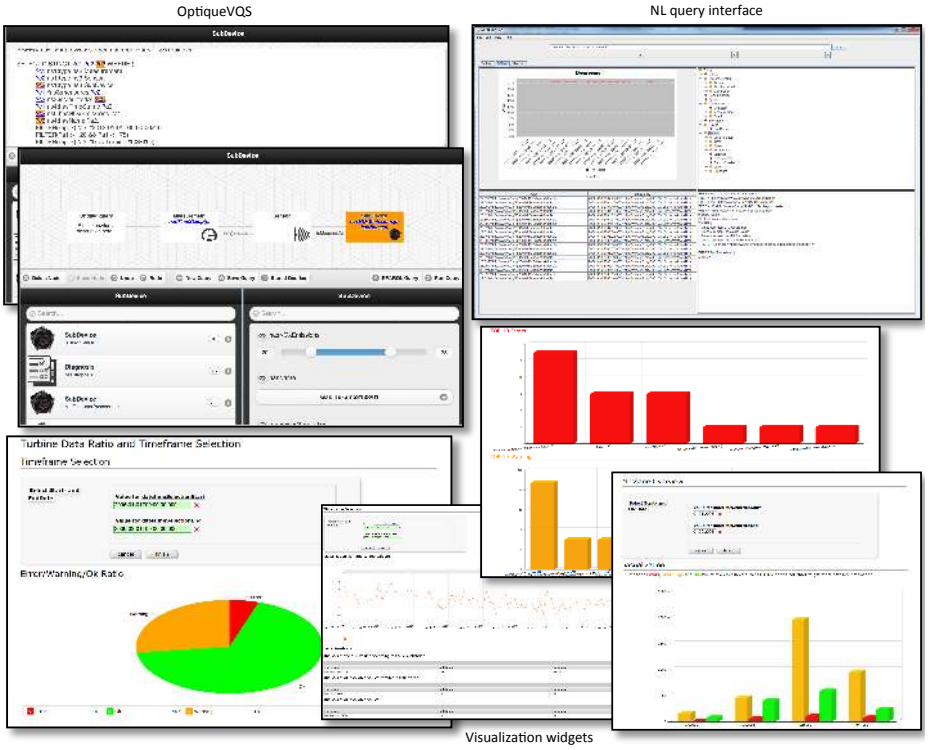

Fig. 5. Query formulation and data visualisation interfaces provided by the software demonstrator

of STARQL was inspired by the Siemens use case requirements. The STARQL query language framework and the prototype streaming engine enjoy the following features:

Expressivity. STARQL allows to express typical mathematical, statistical, and event pattern features needed in real-time monitoring scenarios. In spite of its expressivity, answering STARQL queries is still efficient since they can be transformed into relational stream queries.

Neat Semantics. STARQL comes with a formal syntax and semantics. The latter one uses certain answer semantics [7] and on top of it, first-order logic semantics as in model checking, thereby combining open and closed-world reasoning. A snapshot semantics for window operators [36] is extended with a sequencing semantics that can handle integrity constraints such as functionality assertions.

Orthogonality. 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.

Scope Locality. While producing a SPARQL query, one can select an ontology and streams over which the query will be evaluated. This feature can be important in different cases, e.g., in the case of failure testing, where one is interested in querying only the streams stemming from sensors which are (or are not) suspected to be broken.

Library Functions. Often-used query patterns can be stored in the special library and re-used during query construction.

Now we illustrate the STARQL framework by example.

Example 5. Consider the preventive monitoring request from Example 1. To fulfill it, the following sub-task should be performed: "*Detect a monotonic increase from the temperature sensor*". We now see how this detection can be done within the STARQL framework.

First, assume that the data stream `S_Msmt` is being received from the sensor; its sub-stream that contains data received during the first five seconds is as follows:

$$\{\{\texttt{s0 :val 90}\}\texttt{<0s>}, \{\texttt{s0 :val 93}\}\texttt{<1s>}, \{\texttt{s0 :val 94}\}\texttt{<2s>},$$
$$\{\texttt{s0 :val 92}\}\texttt{<3s>}, \{\texttt{s0 :val 93}\}\texttt{<4s>} , \{\texttt{s0 :val 95}\}\texttt{<5s>}\}. \quad (1)$$

This data is in the form of timestamped RDF triples. For example, the first triple $\{\texttt{s0 :val 90}\}\texttt{<0s>}$ says that at the time point "`0s`" the sensor `s0` sent the value `90`.

Consider the following STARQL query fulfilling the task:

```
CREATE    STREAM S_out_1 AS
SELECT    {s0 rdf:type RMInc}<NOW>
FROM      S_Msmt [NOW-2s, NOW] -> 1s
SEQUENCE  BY StdSeq AS SEQ
HAVING    FORALL i < j  IN SEQ, ?x, ?y:
          IF {s0 :val ?x}<i> AND {s0 :val ?y}<j> THEN ?x <= ?y
```

Intuitively, the structure of the query is as follows:

- The `HAVING` clause specifies that the sensor's value should monotonically increase.
- The `FROM` clause tells that the query performs its check every second, considering only the data from the stream `S_Msmt` in the last two seconds.
- The `SEQUENCE BY` clause groups the output triples using some standard method `StdSeq`.
- The `CREATE` clause declares the query's output stream `S_out_1`.
- The `SELECT` clause determines the format of the timestamped RDF triples in the output stream. For instance, the output stream corresponding to the input data stream from Equation (1) is

  $$\{\{\texttt{s0 rdf:type RMInc}\}\texttt{<0s>}, \{\texttt{s0 rdf:type RMInc}\}\texttt{<1s>},$$
  $$\{\texttt{s0 rdf:type RMInc}\}\texttt{<2s>}, \{\texttt{s0 rdf:type RMInc}\}\texttt{<5s>}\},$$

  where `RMInc` stands for Resent Monotonic Increase, so the timestamped RDF triple $\{\texttt{s0 rdf:type RMInc}\}\texttt{<2s>}$ designates that the sensor `s0` has been experiencing a monotonic increase for the last two seconds, from `0s` to `2s`.

Under the OBDA approach, data is stored in databases, and not in RDF format. Hence, the STARQL engine operates on a virtual stream induced by mappings from the stream `S_Msmt`. In Optique, we are going to use a stream extended version of ADP [23]. ■

5 Demonstrating Capabilities of Extended OBDA at Siemens

In order to demonstrate the potential of OBDA in enhancing data access in Siemens we did a preliminary deployment of the Optique OBDA platform over Siemens data and conducted a preliminary user study. We now give details of our experience.

5.1 Demonstration System

We customised the Optique platform for Siemens and extended it with several components. The demonstration runs on a server with four 8-core Intel64 processors, 512 GB RAM, and 7 TB of internal and 24 TB of external storage.

Data for the demonstration. We installed the platform over databases with 3 TB of historical sensor and event data about 200 gas and steam turbines (15 GB per turbine) gathered between 2002 and 2011. We also developed a data stream generator that simulates the original sensor measurements and events streaming from these turbines.

Demonstration ontology. Although there are ontologies describing machinery with sensors, e.g., the Semantic Sensor Network (SSN) ontology [37], we could not use them: for our use case, they are too generic and overloaded with irrelevant terms, moreover, they miss required terms. Therefore, we constructed our ontologies being guided by the best practices of the SSN ontology. Our ontologies characterize Siemens database schemata of sensor and event data and abstract away from representations varying across data sources. Moreover, our ontologies are manually enriched with the domain information encoded in multiple semiformal and informal models available at Siemens. We developed three ontologies: *(i)* the turbine, *(ii)* sensor, and *(iii)* diagnostic ontology.

The *turbine ontology* describes the internal structure of a turbine, i.e., it lists all its parts, functional units, and their hierarchy. For example, it models that every `Turbine` must have a `ControlSystem` and a `Generator`, and that `LiquidFuelPump` is a part of a `LubOilSystem`. The ontology contains 60 classes and 15 object and data properties. There are three central classes in this ontology: *(i)* `Turbine` class for modeling product families of appliances, *(ii)* `Component` for modeling a hierarchy of subclasses defining the types of components that turbines are constructed of, using relations such as `hasPart` and `hasDirectPart`, *(iii)* `FunctionalUnit` for defining functional interrelation of components, i.e., important blocks of an appliance, such as `GasPath`, `FuelSystem` and others, as well as components belonging to these functional units.

The *sensor ontology* lists and categorizes types of sensors and measuring devices mounted in a turbine as well as their deployment, measurements properties, such as accuracy and precision, and other related information. For example, it models that each sensor is mounted at some turbine's component or functional unit, or that a sensor of a specific type does only produce observations of a given type. The ontology contains 40 classes and 20 properties. The main class `Sensor` covers all types of measuring devices, e.g., `GasDetector`, `TemperatureSensor`. Further branching on classes gives more detailed characteristics information on them, e.g., temperature sensors could measure: `BurnerTipTemperature`, `InletTemperature`, `CompressorExitTemperature`, etc.

The *diagnostics ontology* formalizes relationships between measurements and events generated in by turbine's sensors and control units as well as typical symptoms of faults in turbines. For example, it models that each diagnosis has to be assigned to a turbine or its component, and must be supported by some symptoms. The ontology contains 30 classes and 10 properties. The core classes are `Observation` and `Diagnosis` connected with a relation `indicatesAt` for listing symptoms for each diagnosis.

Each of the three ontologies can be used independently and we also developed an ontology that integrates the three. The turbine and sensor ontologies are expressed in OWL 2 QL, a tractable profile of the OWL 2 ontology language and therefore can be used

straightforwardly in our OBDA setting. The diagnostics ontology must be represented in a richer ontology language: OWL 2 DL. An example of a diagnostic axiom that cannot be expressed in OWL 2 QL is: "If `Turbine` has `Failure` F1, then there must be a `Symptom` S1 in `Turbine Component` C1". To support answering diagnostic queries using OBDA, we provided an approximation of the ontology into OWL 2 QL.

5.2 Preliminary Evaluation at Siemens

We used a two-fold approach to assess the system capabilities at Siemens: *(i)* a query catalog constructed based on the functionality requirements, *(ii)* a user workshop conducted with service engineers to get a feedback from the end users.

Query catalogue. Based on interviews of service engineers and their needs on data acquisitions, we constructed a query catalogue with 27 query patterns from reactive and preventive diagnostics of turbines. This catalogue served as a basis for the development of the temporal query language STARQL (see Section 4.3). There are two types of query patterns: *(i)* on events and measurements, and *(ii)* on diagnostics that requires the usage of semantic knowledge. Query patterns of Type (i) allow to aggregate available sensor values and/or events, generate statistics and visualise results as bar charts and graphical models. An example of such query is "Return the TOP 10 errors and warnings for turbines of product family X". Moreover, some of the query patterns require the usage of temporal operators, e.g., "List sensor values and events within a specific time interval, before an event X occurred". Patterns of Type (ii) reveal interdependency and correlations between *(a)* the occurrence of events, *(b)* sensor measurements, and *(c)* occurrence of events and presence of specific sensor values. An example of a diagnostics query is "Which events frequently occur before the specific point in time?".

We analysed the coverage of the query catalog by the STARQL query language. Though within the development of the STARQL query language the focus was on handling streaming data, it also supports purely temporal reasoning on historical data—which is needed for reactive diagnostics. This can be achieved by considering the time window as a static time filter for the relevant time interval (thereby setting the slide to 0) and then to express the needed patterns in the `HAVING` clause. With this approach, currently, approximately 70% of the query patterns in the catalog can be expressed within STARQL, although some of the queries have to rely on calls to external functions from the ADP system, which provides statistical and mathematical functions.

User workshop. We demonstrated the customised Optique platform at the Siemens service centre in Lincoln, UK, and conducted a preliminary evaluation of the system with IT experts and service engineers. The goal of the evaluation was to obtain an initial feedback from the end users which reinforced and guided our further development of the platform. In particular, the end users were asked to assess the system with respect to the requirements of Section 2. We now summarize the feedback.

The users gave a positive feedback on how the proposed solution addressed Requirement R1 on the integrated data access: information from different source is integrated and can be accessed through one ontology and visualised in the diagnostics dashboard. The users provided also a very positive feedback to the query formulation components of Requirement R2 and highlighted that these facilities may greatly simplify and accelerate the process of query construction. In particular, by using the VQS and the NLQ interface the users were able to specify even complex queries from the query catalogue in a very intuitive way, which currently requires extensive SQL know-how as well as

in-depth knowledge about the database schemata. Additionally, component suggestions on refinements and/or generalisations of the terms used in the query, additional terms and constraints helped the users in understanding the querying capabilities of the platform and in constructing queries, which was highly valued by the users. Likewise, we received good comments on the possibility to derive implicit information using the logical reasoning—thereby satisfying Requirement R3. The streaming support defined in Requirement R4 which is addressed by introducing STARQL query framework was highly welcomed as an important feature for realizing predictive maintenance in future. The ontology and mapping deployment and management support of Requirement R5 was evaluated by the users as crucial to keep the diagnostic knowledge up-to-date.

6 Conclusion and Future Work

In this paper we presented OBDA, a promising paradigm to provide a direct end-user access to data, and how OBDA could enhance this access in Siemens. We derived five requirements that an OBDA solution should fulfill in order to be deployed in Siemens and showed that, while the previous research and system development established the theoretical basis and demonstrated viability of OBDA, a number of limitations have to be solved before industrial deployment of such systems. We then presented an OBDA solution developed within the Optique project that satisfies the five Siemens requirements. We focused on two aspects of the platform specific for Siemens: *(i)* visualisation dashboard and *(ii)* the support of temporal and streaming queries. We also presented preliminary evaluation of the solution at Siemens where we got a positive feedback.

During the evaluation we determined several items to address on our ongoing work on the platform. One of the core questions is the ontology development: The diagnostics ontology is expressed in OWL 2 DL which is not natively supported by the system. Therefore, ontology approximation techniques are strongly required in order to extensively use the knowledge provided by the domain. Also, current ontologies were constructed manually based on formal as well as informal models of the domain. However, bootstrapping information from database schemata, previously executed queries, and already existing formal models and ontology has to be used to reduce manual efforts. Additionally, a possibility to import existing ontologies is strongly needed to support integration of existing ontologies with the ones manually developed.

Another direction of system improvement is the diagnostics dashboard. During the evaluation at Siemens the users pointed out two functionalities that were missing in the demonstrated system: *(i)* reports that incorporate marketing or business intelligence queries, e.g., “Return all gas turbine of a particular product line sold after 2006”, and *(ii)* query interface feature with provenance of query answers and suggestions on possible query repairs. E.g., if a query returns an empty answer, then the users would like to know why the answer set is empty and how the query can be reformulated to obtain answers. Furthermore, to reach 100% coverage of query catalogue we plan to extend functionality of STARQL. We also work on improving the support of temporal and streaming queries.

Integrating the platform into the Siemens IT environment is another important challenge we have to address during future work. The solution has to be adapted in order to be used with the existing Siemens systems for monitoring and analytics. Also, scaling up the solution for the usage in Siemens monitoring environment leads to the integration of multiple streams and databases. Thus, there is a need for distributed query processing and there is an active work within the Optique project in this direction.

7 References

[1] A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012, pp. I–XVIII, 1–497.
[2] M. R. Kogalovsky. Ontology-Based Data Access Systems. In: *Programming and Computer Software* 38.4 (2012).
[3] I. Horrocks. What Are Ontologies Good for? In: *Evolution of Semantic Systems*. Springer, 2013, pp. 175–188.
[4] M. Giese, D. Calvanese, I. Horrocks, et al. Scalable End-user Access to Big Data. In: *Big Data Computing*. Chapman and Hall/CRC, 2013.
[5] E. Kharlamov, E. Jiménez-Ruiz, D. Zheleznyakov, et al. Optique: Towards OBDA Systems for Industry. In: *ESWC (Satellite Events)*. 2013, pp. 125–140.
[6] D. Calvanese, M. Giese, P. Haase, et al. Optique: OBDA Solution for Big Data. In: *ESWC (Satellite Events)*. 2013.
[7] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking Data to Ontologies. In: *J. Data Semantics* 10 (2008), pp. 133–173.
[8] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. In: *J. of Automated Reasoning* 39.3 (2007), pp. 385–429.
[9] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, and D. F. Savo. The MASTRO System for Ontology-Based Data Access. In: *Semantic Web* 2.1 (2011), pp. 43–53.
[10] F. Priyatna, O. Corcho, and J. Sequeda. Formalisation and Experiences of R2RML-based SPARQL to SQL Query Translation Using Morph. In: *WWW*. 2014.
[11] M. Rodriguez-Muro, R. Kontchakov, and M. Zakharyaschev. OBDA with Ontop. In: *ORE*. 2013, pp. 101–106.
[12] C. Bizer and A. Seaborne. D2RQ-Treating non-RDF Databases as Virtual RDF Graphs. In: *ISWC*. 2004.
[13] K. Munir, M. Odeh, and R. McClatchey. Ontology-Driven Relational Query Formulation Using the Semantic and Assertional Capabilities of OWL-DL. In: *Knowl.-Based Syst.* 35 (2012), pp. 144–159.
[14] J. F. Sequeda and D. P. Miranker. Ultrawrap: SPARQL Execution on Relational Data. In: *J. of Web Sem.* 22.0 (2013).
[15] A. Tian, J. Sequeda, and D. P. Miranker. QODI: Query as Context in Automatic Data Integration. In: *ISWC*. 2013.
[16] M. G. Skjæveland, E. H. Lian, and I. Horrocks. Publishing the Norwegian Petroleum Directorate's FactPages as Semantic Web Data. In: *ISWC*. 2013, pp. 162–177.
[17] E. Kharlamov, M. Giese, E. Jiménez-Ruiz, et al. Optique 1.0: Semantic Access to Big Data: The Case of Norwegian Petroleum Directorate's FactPages. In: *ISWC (Posters & Demos)*. 2013.
[18] P. Haase, I. Horrocks, D. Hovland, et al. Optique System: towards Ontology and Mapping Management in OBDA Solutions. In: *WoDOOM*. 2013, pp. 21–32.
[19] B. Glimm, I. Horrocks, B. Motik, and G. Stoilos. Optimising Ontology Classification. In: *ISWC*. 2010, pp. 225–240.
[20] D. Calvanese, I. Horrocks, E. Jiménez-Ruiz, E. Kharlamov, M. Meier, M. Rodriguez-Muro, and D. Zheleznyakov. On Rewriting, Answering Queries in OBDA Systems for Big Data. In: *OWLED*. 2013.
[21] H. Kllapi, D. Bilidas, I. Horrocks, Y. E. Ioannidis, E. Jiménez-Ruiz, E. Kharlamov, M. Koubarakis, and D. Zheleznyakov. Distributed Query Processing on the Cloud: the Optique Point of View. In: *OWLED*. 2013.
[22] I. Horrocks, T. Hubauer, E. Jiménez-Ruiz, et al. Addressing Streaming and Historical Data in OBDA Systems: Optique's Approach. In: *KNOW@LOD*. 2013, pp. 33–40.
[23] M. M. Tsangaris, G. Kakaletris, H. Kllapi, G. Papanikos, F. Pentaris, P. Polydoras, E. Sitaridi, V. Stoumpos, and Y. E. Ioannidis. Dataflow Processing and Optimization on Grid and Cloud Infrastructures. In: *IEEE Data Eng. Bull.* 32.1 (2009), pp. 67–74.
[24] P. Haase, C. Hütter, M. Schmidt, and A. Schwarte. The Information Workbench as a Self-Service Platform for Linked Data Applications. In: *WWW*. 2012.
[25] A. Soylu, M. Giese, E. Jimenez-Ruiz, E. Kharlamov, D. Zheleznyakov, and I. Horrocks. OptiqueVQS – Towards an Ontology-Based Visual Query System for Big Data. In: *MEDES*. 2013.
[26] A. Soylu, M. G. Skjæveland, M. Giese, I. Horrocks, E. Jiménez-Ruiz, E. Kharlamov, and D. Zheleznyakov. A Preliminary Approach on Ontology-Based Visual Query Formulation for Big Data. In: *MTSR*. 2013, pp. 201–212.
[27] U. Waltinger, D. Tecuci, M. Olteanu, V. Mocanu, and S. Sullivan. Natural Language Access to Enterprise Data. In: *AI Magazine* 35.1 (2014), pp. 38–52.
[28] A. Artale, R. Kontchakov, F. Wolter, and M. Zakharyaschev. Temporal Description Logic for Ontology-based Data Access. In: *IJCAI*. 2013, pp. 711–717.
[29] S. Borgwardt, M. Lippmann, and V. Thost. Temporal Query Answering in the Description Logic DL-Lite. In: *FroCoS*. 2013, pp. 165–180.
[30] D. F. Barbieri, D. Braga, S. Ceri, E. D. Valle, and M. Grossniklaus. C-SPARQL: a Continuous Query Language for RDF Data Streams. In: *Int. J. Semantic Computing* 4.1 (2010), pp. 3–25.
[31] J.-P. Calbimonte, O. Corcho, and A. J. G. Gray. Enabling Ontology-Based Access to Streaming Data Sources. In: *ISWC*. 2010, pp. 96–111.
[32] D. L. Phuoc, M. Dao-Tran, J. X. Parreira, and M. Hauswirth. A Native and Adaptive Approach for Unified Processing of Linked Streams and Linked Data. In: *ISWC*. 2011, pp. 370–388.
[33] Y. Zhang, P. Minh Duc, O. Corcho, and J. P. Calbimonte. SRBench: A Streaming RDF/SPARQL Benchmark. In: *ISWC*. 2012, pp. 641–657.
[34] Özgür L. Özçep, R. Möller, C. Neuenstadt, D. Zheleznyakov, and E. Kharlamov. *Deliverable D5.1 – A Semantics for Temporal and Stream-Based Query Answering in an OBDA Context*. Deliverable FP7-318338. EU, 2013.
[35] Özgür L. Özçep, R. Möller, and C. Neuenstadt. *OBDA Stream Access Combined with Safe First-Order Temporal Reasoning*. Technical Report. Available at `http://www.sts.tuhh.de/tech-reports/papers.html`: Hamburg University of Technology, 2014.
[36] A. Arasu, S. Babu, and J. Widom. The CQL Continuous Query Language: Semantic Foundations and Query Execution. In: *The VLDB Journal* 15.2 (2006), pp. 121–142.
[37] M. Compton, P. M. Barnaghi, L. Bermudez, et al. The SSN Ontology of the W3C Semantic Sensor Network Incubator Group. In: *J. Web Sem.* 17 (2012), pp. 25–32.