Semantic Web 1 (2016) 1–28
IOS Press

OptiqueVQS: a Visual Query System over Ontologies for Industry [1]

Editor(s): Name Surname, University, Country
Solicited review(s): Name Surname, University, Country
Open review(s): Name Surname, University, Country

Ahmet Soylu [a,*], Evgeny Kharlamov [b], Dmitriy Zheleznyakov [b], Ernesto Jimenez-Ruiz [b], Martin Giese [c], Martin G. Skjæveland [c], Dag Hovland [c], Rudolf Schlatte [c], Sebastian Brandt [d], Hallstein Lie [e] and Ian Horrocks [b]

[a] *NTNU – Norwegian University of Science and Technology, Gjøvik, Norway*
E-mail: ahmet.soylu@ntnu.no
[b] *Department of Computer Science, University of Oxford, Oxford, UK*
E-mail: {name.surname}@cs.ox.ac.uk
[c] *Department of Informatics, University of Oslo, Oslo, Norway*
E-mail: {martingi, martige, hovland, rudi}@ifi.uio.no
[d] *Corporate Technology, Research and Technology Center, Siemens AG, Munich, Germany*
E-mail: sebastian-philipp.brandt@siemens.com
[e] *Statoil ASA, Stavanger, Norway*
E-mail: hali@statoil.com

Abstract. An important application of semantic technologies in industry has been the formalisation of information models using OWL 2 ontologies and the use of RDF for storing and exchanging application data. Moreover, legacy data can be virtualised as RDF using ontologies following the Ontology-Based Data Access (OBDA). In all these applications, it is important to provide domain experts with query formulation tools for expressing their information needs over ontologies. In this work, we present such a tool, OptiqueVQS, that has been designed based on our experience with OBDA applications in Statoil and Siemens and on the best HCI practices for interdisciplinary engineering environments. OptiqueVQS implements a number of unique techniques that distinguish it from analogous query formulation systems. In particular, it exploits ontology projection techniques to enable graph-based navigation over an ontology during query construction time. Secondly, while OptiqueVQS is primarily ontology driven, it exploits sampled data to enhance selection of data values for some data attributes. Finally, OptiqueVQS is built on well grounded requirements, design rationale, and quality attributes. We have evaluated OptiqueVQS with both domain experts and casual users and qualitatively compared our system against prominent visual systems for ontology-driven query formulation and exploration of semantic data. OptiqueVQS is available online and can be downloaded together with an example OBDA scenario.

Keywords: visual query formulation, OWL 2 ontologies, RDF data, SPARQL queries, data retrieval, usability

1. Introduction

Adoption of semantic technologies has been a recent development in many large companies such as IBM [33], the steel manufacturer Arcelor Mittal [6], the oil and gas company Statoil [50], and Siemens [53, 3,65]. An important application of these technologies

[1] This work was funded by the EU FP7 Grant "Optique" (agreement 318338), and by the EPSRC projects MaSI[3], DBOnto, and ED[3].
[*] Corresponding author. E-mail: ahmet.soylu@ntnu.no

has been the formalisation of information models using OWL 2 ontologies and the use of RDF for storing application data. OWL 2 provides a rich and flexible modelling language that well-suited for describing industrial information models [49,34,2]: it not only comes with an unambiguous, standardised, semantics, but also with a wide range of tools that can be used to develop, validate, integrate, and reason with such models. In turn, RDF data can not only be seamlessly accessed and exchanged, but also stored directly in highly scalable RDF triple stores and effectively queried in conjunction with the available ontologies. Moreover, legacy and other data that must remain in its original format and cannot be transformed into RDF can be virtualised as RDF using ontologies following the Ontology-Based Data Access (OBDA) approach [48,53,23,81,54,63].

In all these applications, it is important to provide domain experts—who has extensive domain knowledge but not necessarily skills and knowledge in semantic technologies and formal query languages such as SPARQL—with query formulation tools for expressing their information needs over ontologies. The problem of query formulation for end users has been acknowledged by many [22,35,7,50,72,75] and numerous systems have been developed so far. These systems can be categorised as follows:

1. *Textual query editors* (e.g., Virtuoso[1]) employ the full expressivity of SPARQL, but demand technical skills and knowledge (i.e., on syntax and schema). Context-aware editors, such as SparQLed [20], offer auto-completion and recommendations based on the schema and dataset.
2. *Keyword search* (e.g., [15]) interprets a query as a bag of words. They are simple to use, but are inherently limited in expressiveness. There exists approaches, such as KESOSD [58] and SWSE [41], aiming at increasing the accuracy and completeness of keyword search.
3. *Natural language interfaces* (e.g., [46,57]) interpret a query as whole and take linguistic considerations into account, but suffer from ambiguities and linguistic variability. There are approaches to overcome this problem, such as user dialogues for feedback and clarification [26].
4. *Visual query languages* (VQL), such as RDF-GL [42] and QueryVOWL [37], are based on a well-defined formal semantics with a visual notation and syntax. They are comparable to formal textual languages as they demand high technical skills and knowledge.
5. *Visual query systems* (VQS) (cf. [22]), such as Rhizomer [16] and Konduit VQB [5], are based on a system of interactions, rather than a visual formalism, therefore demand no technical background. They often compromise expressivity to reach a fine expressiveness and usability balance.

To the best of our knowledge none of such systems has been developed upon industrial requirements or evaluated with industrial users. In this work we present a visual query formulation system, OptiqueVQS [80, 74], that has been designed upon *(i)* requirements from Statoil and Siemens that we consolidated during a joint OBDA project, called Optique[2] [31,30], with these companies and *(ii)* best HCI practices for interdisciplinary engineering environments.

OptiqueVQS implements a number of unique techniques that distinguish it from analogous query formulation systems. In particular:

- it exploits ontology projection techniques to enable graph-based navigation over an ontology during query construction time;
- while OptiqueVQS is primarily ontology driven, it exploits sampled data to enhance selection of data values for some data attributes;
- it is built on well grounded requirements, design rationale, and quality attributes;
- and it has been evaluated with different types of end users in different contexts.

We evaluated OptiqueVQS with different user groups and contexts: a study involving casual users [74]; a comparative study with PepeSearch [90]; and three studies, which are reported in this article, with Statoil and Siemens domain experts. Our studies provided encouraging results; in particular, studies with Statoil and Siemens users revealed that domain experts could use OptiqueVQS to formulate queries meeting their daily data needs in a few minutes with high effectiveness.

Finally, we qualitatively compared OptiqueVQS against prominent existing visual systems for ontology-driven query formulation and exploration of semantic data that are the most relevant to our system. For

[1] Virtuoso SPARQL Query Editor `http://dbpedia.org/sparql`

[2] Optique project: `http://optique-project.eu`

the comparison we considered gFacet [40], OZONE [82], SparqlFilterFlow [36], Konduit VQB [5], and Rhizomer [16], PepeSearch [89], Super Stream Collider framework [64], and TELIOS Spatial [28]. The comparison revealed that OptiqueVQS possesses an important set of quality attributes relevant in an industrial context, while others meet only a few of them. OptiqueVQS is available online and it can be downloaded together with an example OBDA scenario, including a data set, an ontology, mappings etc., form the project's website (see Section 5 for details).

The rest of article is organised as follows: in Section 2 we present preliminary notations and concepts used through the article. In Section 3 we present Statoil and Siemens use-cases. In Section 4 we discuss a set of requirements and quality attributes, while in Section 5, we present OptiqueVQS. In Sections 6 and 7, we evaluate OptiqueVQS first against the requirements and then with a set of usability studies respectively.

This submission extends the previously published material on OptiqueVQS in several important directions. In particular, we present:

- a set of concrete requirements collected through a systematic requirement collection process;
- OptiqueVQS extensions for spatial and temporal query formulation support;
- three extensive user studies with domain experts at Siemens and Statoil;
- a qualitative comparison with eight other query formulation systems;
- and an improved OptiqueVQS' backend and a more detailed description of it, including its expressive power.

2. Preliminaries

In the following we give a brief tour through some notions from Description Logics (DL), RDF, OWL, SPARQL queries and their semantics. The goal of this section is to semi-formally introduce some relevant Semantic Web notions that we follow, and to make the reader ready to examples, and explanations that appear in the article and to use the DL syntax. Since in this article we study how to support construction of queries over ontologies and data in industrial settings, and focus mostly on user driven requirements rather than complexity and other formal accounts, we make the formal descriptions below light weight, and refer the reader to relevant material for more details.

We use standard notions from first-order logic. We assume pairwise disjoint countably infinite sets of *constants*, *unary predicates*, also called *(atomic) classes* and *binary predicates*, also called *properties*. Constants, in turn, are constituted of disjoint sets of *objects or individuals* and *literal values*. We treat $\top$ and $\bot$ as special unary predicates, which are used to represent a tautology and falsehood, respectively. A *fact* is a ground atom and a *knowledge base* is finite set of facts.

An *ontology* is a finite set of first-order sentences. The Web Ontology Language OWL 2 [25] is a recursive set of ontologies, closed under renaming of constants and the subset relation. Each OWL 2 ontology can be represented using a specialised DL syntax [9,44] where variables are omitted and which provides operators for constructing complex concepts and properties from simpler ones, as well as a set of axioms. The semantics of an OWL 2 ontology is defined in a standard way using first-order interpretations [1]. Note that, for convenience and readability, in the examples we use the Manchester OWL syntax [43], which is a user-friendly compact syntax for OWL 2 ontologies

For example, consider the following statement about the application domain:

"every wellbore has (at least one) core".

It can be written as an OWL 2 axiom of the form:

Wellbore SubClassOf: hasCore some Core

SPARQL [39] is the standard query language to access RDF data enhanced with OWL 2 ontologies. SPARQL queries $Q(\vec{x})$ are defined in terms of graph patterns, i.e., sets of triples of the form $\langle n_1,\ e,\ n_2\rangle$ that are referred to as basic graph patterns and where n_is denote nodes in a graph and e denotes an edge, and they can either be concrete nodes (i.e., constants or unary predicates) and edges (i.e., binary predicates), or variables; some of these variables form the vector $\vec{x}$ of Q's output variables. In this work we focus on construction of SPARQL queries where basic graph patterns do not have variables on the second position, nor on the third position, when e is rdf:type. That is, we do not allow predicates as variables, and thus our queries can naturally be represented as conjunctions of unary and binary atoms. SPARQL 1.1 also allows for the union of graph patterns, aggregate functions, and other operators. In our work we focus on construction of conjunc-

tive queries with aggregation and distinct operators. The semantics of query answering is defined in the standard way in terms of homomorphism [4]: a vector of constants $\vec{t}$ is the answer for a conjunctive query $Q(\vec{x})$ over a dataset D, if there is a homomorphism from the query to D such that the vector of output variables is matched to $\vec{t}$. This semantics can be naturally extended from datasets to first-order logic (FOL) interpretations of datasets and ontologies [9].

3. Industrial Use Cases

In the context of the Optique project, we have used use cases from Statoil and Siemens including sample queries and data sets to feed the development of OptiqueVQS and later to evaluate it. We believe that they are representative for many of the data access challenges faced by today's data-intensive industries.

Statoil and Siemens have their data stored in relational databases rather than triple stores, as majority of world's enterprises do. In the Optique project, the use case data sets have been represented as a knowledge bases using an ontology-based data access (OBDA) technology to enable in-place querying of legacy relational data sources [31]. OBDA technologies are important in the context of visual query formulation as well, as they extend the reach of ontology-based visual query formulation from triple stores to relational databases; hence, raising it as a viable and realistic solution for all. The OBDA approach we employed is built on two mechanisms [81,19]:

(a) *mappings* are used to virtualise the relational data in databases into graph data expressed over a language defined in an ontology;
(b) and *query rewriting* is used expand and translate the posed queries (e.g., in SPARQL) into the language of the underlying relational database system (e.g., to SQL).

The complete details of underlying OBDA framework is out of scope of this work; therefore, we refer interested readers to the Optique project [30,31,53,50, 52].

In the following subsections, we describe the characteristics of each use case. Descriptions were provided by the organizations themselves and confirmed through interviews and on-site visits. We highlight and mark some parts of the descriptions to support requirements derived in Section 4 (i.e., ^{En}). We are often not able to disclose the exact numbers in the descriptions due to the privacy policies of Statoil and Siemens.

3.1. Statoil Use Case

The overall goal of the Statoil use case in the Optique project is to enable *geoscientists at Statoil to find answers to their own information needs* (E1) —questions that generally concern locating new petroleum deposits. Domain experts with technical data science skills and knowledge are rare. Database schemas in use are often designed from an abstract generic information model and presents itself quite obscurely to its end-users. Building interesting SQL queries *require therefore in many cases a very large number of table joins (i.e., 20 to 30 tables)* (E2), thus making the task of handcrafting SQL queries towards this database very complex and time-consuming.

To access the data sets, Statoil personnel use special purpose software tools that contain predefined and mostly generic queries. The data sets are never directly accessed by domain experts through handcrafted queries. Hence, in order to answer specific and detailed information needs a Statoil domain expert must gather data from the answer sets of multiple such predefined queries and process the answers by manipulating, joining and filtering the data in other software tools, like spreadsheet applications. This is a manual task that is prone to error, inefficient and is difficult to automate and reproduce. Moreover, the database extraction tool is complex and *contains a large number of predefined queries* (E3), so finding the correct queries can be an elaborate process, and due to the complexity of the tool and the underlying database schema new queries are in practice never added to the tool. *Domain experts spend considerable time on data extraction activities daily* (E4); and therefore, in this situation, the value creation potential is severely limited.

3.2. Siemens Use Case

Siemens runs several service centres for power plants, each responsible for remote monitoring and diagnostics of many thousands of gas/steam turbines and associated components such as generators and compressors. *Diagnosis engineers working at the service centres* (E5) are informed about any potential problem detected on site. Unlike Statoil, a good number of diagnosis engineers at Siemens have technical skills and knowledge. They access a variety of raw and processed data with pre-defined queries in order to isolate the problem and to plan appropriate maintenance activities. For diagnosis situations not initially anticipated, new queries are required, and an IT expert familiar

with both the power plant system and the data sources in question has to be involved *to formulate various type of queries* (E6). Thus, unforeseen situations may lead to significant delays of up to several hours or even days.

The required data is *spread over hundreds of tables with very complex structure for event data* (E7) (e.g., up to 2.000 sensors in a part of appliance and static data sources). With few built-in features for manipulating time intervals, traditional data base systems often offer insufficient support for querying time series data, and it is highly non-trivial to combine querying techniques with the statistics-based methods for trend analysis that are typically in use in such cases. *Domain experts' daily routine are very data intensive* (E8), and with the ability to formulate complex queries on their own with respect to an expressive and high-level domain vocabulary, IT experts will not be required anymore for adding new queries, and manual pre-processing steps can be avoided.

4. Requirements

Domain experts have an in-depth knowledge and understanding of the semantics of their expertise domain. However, they might or might not have technical skills and knowledge such as on programming, databases, query languages. In the latter case, they often have low tolerance, intension, or time to use and learn formal textual query languages. Therefore, our primary goal is to provide a visual query specification mechanism for users who otherwise cannot or do not desire to use formal textual query languages to retrieve data. We also expect that domain experts with technical skills and knowledge could often benefit from the availability of such visual mechanism, particularly if they are given the opportunity to switch between textual and visual query formulation within a task.

Visual query formulation (cf. [22,72,75]), as an *end-user development* paradigm (cf. [56]), is promising to remediate end-user data access problem. It is built on the *direct manipulation* idea [67], in which end users recognise and interact with the visual representations of domain elements, rather than recalling domain and syntax elements and programmatically combining them. Visual approaches for query specification could be considered in two categories [29]. First category refers to *visual query systems* (VQSs); a VQS is a system of interactions built on an informal set of user actions that effectively capture a set of syntactic rules specifying a query language (e.g, [5,40]). Second category refers to *visual query languages* (VQLs); a VQL is a combination of formal visual notation and syntax representing the semantics and syntax of a query language (e.g., [12,37]). A VQL is as difficult as a formal textual query language for a domain expert as it demands considerable technical skills and knowledge to interpret the visual semantics and syntax and understand the relevant technical jargon.

A VQS has to support certain data access efforts: *exploration*, i.e., understanding the reality of interest, which relates to the activities for understanding and finding schema concepts and relationships relevant to information need at hand; and *construction*, which concerns the compilation of relevant concepts and constraints into formal information needs (i.e., queries) [22]. On these grounds, the choice of *visual representation* and *interaction paradigms*, along with underlying metaphors, analogies etc., is of primary importance. Catarci et al. [22] classify VQSs with respect to used visual representation paradigms, such as forms, diagrams, and icons, and interaction paradigms, such as navigation and browsing. The choice of appropriate representation and interaction paradigm depends on query, task, and user types, such as the variance of query tasks, the structural complexity of queries, and users' familiarity with the subject domain [22].

One should also realise the distinction between *browsing* and *querying*. In the former users, to a large extent, operate at data level to filter down an information space, e.g., faceted search interfaces (e.g., [86]). In the latter, which we predominantly use in OptiqueVQS, users have direct interaction with the vocabulary of the domain (concepts and relations) (e.g., [12]), but not directly with concrete data as e.g., in OLAP cube interfaces. This is necessary because:

(a) the queries we need to pose are more complex than what can be achieved by more data oriented interfaces;
(b) and both the evaluation of those queries and the caching of all possible precomputed results would use too much resources.

Query formulation is a complex task for domain experts and other non-skilled users; therefore, an end-user visual query formulation tool is often limited in expressiveness to ensure a good usability. End users make a very little use of advanced functionalities and likely to drop their own requirements for the sake of having simpler ways for basic tasks [21]. A VQS at

right level of expressiveness does not necessarily mean that it will be adopted by end users and organisations, unless it reaches to a certain level of quality in terms of user experience, system design, and run-time performance.

Overall, we highlight three main challenges:

(C1) Identifying common query types (i.e., typicality), which are reasonably complex (i.e., perceived complexity) and would meet the majority of end-user tasks in order to set an appropriate balance between usability and expressiveness.
(C2) Identify query, task and user types at hand in order to select representation and interaction paradigms that fit best.
(C3) Identify a set of *quality attributes* (cf. [47]), i.e., non-functional requirements, ensuring that a VQS can function and evolve as needed.

Accordingly, in the following, we list an elaborate on a set of requirements in terms of expressivity and quality attributes.

4.1. Expressiveness

In order to address C1, we first studied the typicality by constructing a query catalogue from the 97 representative sample queries provided by Statoil in natural language. Information needs in the query catalogue are considered as patterns of information needs, and each such request represents one topic that geologists are typically interested in. We verified with domain experts that the catalogue provides a good coverage for the information needs of Statoil geologists.

Two SPARQL experts reformulated these information needs in SPARQL given a domain ontology. Then we made a syntactical analysis of the query catalogue (see Figure 1) with respect to notable query types described in Table 1 and with respect to SPARQL specification [39]. These query types include conjunctive queries (QT1), disjunctive queries (QT2), queries with cycle (QT3), queries with aggregation (QT4), queries with negation (QT5), and ground queries (QT6). The identification of queries of QT1, QT2, QT4 and QT5 are straight forward as they are built on clear SPARQL operators; however identification of QT3 queries are more involved as it relates to the topology of a given query. Therefore, we transformed each query into an undirected-labelled graph and executed a cycle detection algorithm to identify QT3 queries.

The analysis suggest that majority of end-user queries are ground queries, i.e., 64 % (see Figure 1 (a)). At this point we considered potential perceived complexity by the end users. Queries that involve cycles are more difficult to formulate as they involve visiting the same node twice. Queries that involve disjunction and negation, particularly at object property level, are also comparatively difficult as they would require a deeper understanding of these notions. Therefore, we are led to the first requirement that:

(R1) Support the formulation of tree-shaped conjunctive queries.

In order to address C2, we have conducted a thorough conceptual literature survey [75]. Particularly the longstanding literature on visual query formulation over relational databases reveals a substantial amount of findings [22]. We employed a framework suggested by Catarci et al. [22] and considered dimensions presented in Table 2 to identify suggested paradigms for each dimension in the order of priority. All queries in the query catalogue are unique and cover a wide range of typical information needs. The query catalogue shows that (Figure 1 (b)) 73 % of queries involve more than 3 concepts referring to a high structural complexity. These evaluations led us to the second requirement:

(R2) Provide a multi-paradigm user-interface where a diagram-based paradigm has central role and is supported by forms-based and iconic representation paradigms.

This requirement is inline with one global finding that visual query tools that combine multiple representation and interaction paradigms are better to address varying user, task, and query types [22,45].

In Siemens case, a query catalogue does not exist due to a higher privacy policy. However, it is verified by the domain experts of Siemens that their information needs follow the characteristics of Statoil's query catalogue. One should note that the Siemens case focuses on streaming sensor data (i.e., temporal), which leads to somewhat more domain-specific requirements on the user interface – basically possibility to involve stream properties and to select relevant stream templates and parameters. This is also partly valid for Statoil as they often deal with geographical data (i.e., spatial) and domain experts would benefit a lot from a map component for constraining and selecting data values. Therefore, a third requirement also needs to be met:

(R3) Provide domain-specific components for dealing with temporal and spatial data sources.

Table 1
Description of query types.

#	Query type	Description	SPARQL syntax
QT1	Conjunctive queries	Query atoms in a given query are connected only with AND connective.	SPARQL queries with basic graph pattern (BGP) and group graph patterns (GGP).
QT2	Disjunctive queries	Some query atoms in a given query are connected with OR connective.	SPARQL queries with multiple optional graph patterns (MOGP), i.e. OPTIONAL, and alternative graph patterns (AGP), i.e., UNION.
QT3	Queries with cycle	Query graph includes at least one path where a node is visited twice.	SPARQL queries having at least one path which start and end with the same node, when the query graph is viewed as undirected labelled graph and type assertions are omitted.
QT4	Queries with aggregation	The values of multiple output elements are grouped together to form a single value.	SPARQL queries including aggregate functions such as MIN, MAX, AVG, SUM.
QT5	Queries with negation	Queries that involve checking whether certain triples don't exist in the data graph.	SPARQL queries that involve negation by failure through NOT EXISTS, MINUS, NOT IN, and !BOUND operators.
QT6	Ground queries	Queries that are conjunctive and three-shaped and do not include negation and aggregation.	SPARQL queries that are conjunctive and do not include cycle, aggregation, and negation.

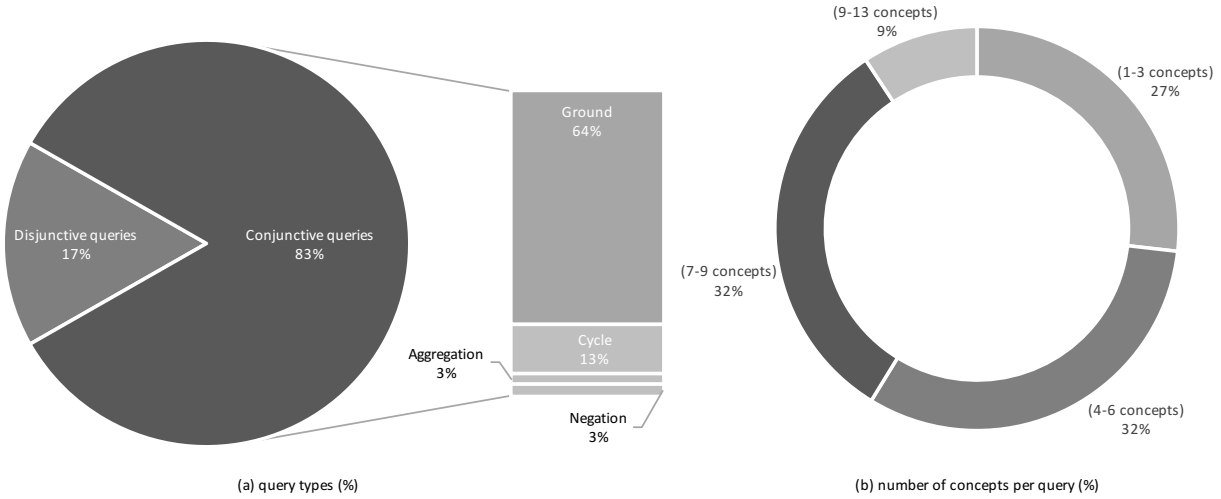

Fig. 1. An analysis of the Statoil query catalogue.

Table 2
Framework for selecting the representation paradigms.

Dimension	Level	Support	Suggested paradigms
Frequency of interaction	Frequent	Uses cases (E4, E8)	1. form-based and 2. diagram-based
Variance of query tasks	Extemporary	Use cases (E3, E6) and query catalogue (Statoil)	1. icon-based and 2. diagram-based
Structural complexity	Sophisticated	Use cases (E2, E7) and query catalogue (Statoil)	1. diagram-based
Domain familiarity	Familiar	Use cases (E1, E5)	1. diagram-based

4.2. Quality attributes

Quality attributes are non-functional requirements that effect *run-time behaviour*, *design*, and *user expe-*

rience. From an end-user development perspective, we derive a set of quality attributes for VQSs, which effectively increases the benefits gained and decrease the cost of adoption for end users (cf. [83]). We followed the approach employed by Khalili and Auer [47] and extracted a set of quality attributes. For this purpose, we used the conceptual survey we conducted earlier [75,72] as well as input we received from the use case partners. In the following, we describe the attributes, which are relevant in our context.

(A1) **Usability** refers the capacity of a system to meet its identified aims and is measured in terms of its effectiveness (i.e., accuracy and completeness), efficiency (i.e., time/effort required), learnability (i.e., time and effort required to learn the tool), and user satisfaction.

(A2) **Modularity** refers to the degree which a system's components are independent and interlocking. A highly modular system ensures flexibility and extensibility, so that new components could easily be introduced to adapt to changing requirements and to extend and enrich the functionality provided.

(A3) **Scalability** refers to ability of a VQS to visualise and deal with large ontologies in our context. A scalable VQS increases comprehensibility by avoiding the cluttering and scattering of presentation and cognitive overload, which in turn makes formulation and exploration easier against large ontologies.

(A4) **Adaptivity** refers to ability of a system to alter its behaviour, presentation, and content with respect to context. A VQS could reduce the effort required for query formulation by adaptively offering concepts and properties, for instance with respect to previously executed queries (i.e., query log).

(A5) **Adaptability**, in contrary to adaptivity, is a manual process, where users customise a system to their own needs and contexts. An adaptable VQS could provide flexibility against changing requirements, e.g., one can add a new domain-specific representation component.

(A6) **Extensibility** refers to the ability and the degree of effort required to extend a system. An extensible VQS provides flexibility against changing requirements by providing room, from both architectural and design perspectives, for sustainable evolution.

(A7) **Interoperability** refers to the ability of system to communicate and exchange data with other applications. Interoperability contributes to the functionality of a VQS by allowing it to utilise or feed other applications in an organisational workflow or digital ecosystem.

(A8) **Portability** refers to the ability of a VQS to query other domains, rather than only a specific domain, without high installation and configuration costs. Domain-specific components (e.g., presentation modules) could be offered if available; however, the lack of domain-specific components should not be blocking.

(A9) **Reusability** refers to ability of a VQS to utilise queries as consumable resources in our context. Reusability could decrease the learning effort by utilising previous queries for didactic purposes and allow users to formulate more complex queries by modifying the existing ones.

4.3. Discussion

A VQS should not be considered in isolation from the *context*, which could be characterised by a variety of dimensions such as user, task, data, and organisation [27,70]. In this respect, quality attributes presented previously are related and support usability directly/indirectly. They mainly ensure sustainability against potential variances in context, in other words, to support the evolution of a VQS against ever-changing context dimensions without losing the expressiveness-usability balance. For example, the heterogeneity of data necessitates domain-specific presentation and interaction components for improved user experiences. In this respect, modularity and extensibility plays an underpinning role by facilitating the development and integration of such components. Another example would be the organisational context: a VQS is often a part of larger tool portfolio for data extraction, analysis, and decision-making, and in this context interoperability is valuable to ensure a seamless orchestration.

One of the main problems that typical VQS systems face is the scalability against large ontologies (cf. [45]). A VQS has to provide its users with the fragments of ontology (e.g., concepts and properties) continuously, so that users can select relevant ontology elements and iteratively construct their queries. However, even with considerably small ontologies, the number of concepts and properties to choose from increases drastically due to the propagation of property

restrictions [24]. In turn, the high number of ontology elements overloads the user interface and hinders usability (i.e., scattering and cluttering). The aforementioned problem can be approached with gradual access to ontology and adaptivity (cf. [17]) (i.e., selecting and displaying the most relevant fragments of the ontology at each step).

The structural complexity of query tasks deserves a special attention for choosing the right representation and interaction paradigms. Our use cases come with non-simple query tasks, which are structurally complex. Respectively, navigational interaction style becomes essential, i.e., *query by navigation* (QbN) [84,78]. Recent faceted search approaches, which are originally used to browse instances of a single concept, even come with possibility to navigate and combine a number of concepts and create complex structures to retrieve data (e.g., [16,8]). We consider graph-based representation and navigation as an appropriate choice in this respect. This is because graphs are effective mechanisms to navigate, construct, communicate complex topological structures for end users (cf. [22,45]). Secondly, it is well-known that the majority of end-user queries are conjunctive, and thus, in the semantic web setting, they could naturally be seen as graphs since we are dealing with unary and binary predicates only.

5. OptiqueVQS

OptiqueVQS is composed of an interface and a navigation graph extracted from the underlying ontologies. The interface components are populated and driven according to the information in the navigation graph. In the following subsections we present each part.

OptiqueVQS is available online[3] together with the whole Optique platform, a comprehensive tutorial, and an example OBDA scenario including artefacts such as ontology, data set, and mappings for online testing and download[4].

[3]Quick access to OptiqueVQS online demo (username and password: “demo"): `http://optique-northwind.fluidops.net/resource/VisualQueryFormulation`

[4]The whole Optique platfrom with an example OBDA scenario for online testing and download: `http://optique-project.eu/northwind-tutorial/`

5.1. OptiqueVQS frontend

The OptiqueVQS interface is designed as a *widget-based user-interface mashup* (i.e., UI mashup), which aggregates a set of applications in a common graphical space, in the form of *widgets*, and orchestrates them for achieving common goals (cf. [79]). Apart from flexibility and extensibility, such a modular approach provides us with the ability to combine multiple representation and interaction paradigms, and distribute functionality to appropriate widgets.

Initially, three widgets appear in OptiqueVQS, as depicted in Figure 2 (recall R2 at Section 4):

(W1) The first widget is a menu-based QbN widget accompanied with icons and allows the user to navigate concepts by picking relationships between them (see the bottom-left part of Figure 2).

(W2) The second widget is form-based, and presents the attributes of a selected concept for selection and projection operations (see the bottom-right part of Figure 2).

(W3) The third widget is diagram-based, and presents the constructed query and affordances for manipulation (see the top part of Figure 2) .

On the one hand, W1 and W2 provide *view*; i.e., they focus the user to the current phase of the task at hand by providing means for gradual and on demand exploration and construction. On the other hand, W3 provides an *overview*, i.e., an outlook of the query formulated so far, and lets the user to refocus. These three widgets are orchestrated by the system, through harvesting event notifications generated by each widget as the user interacts.

A typical interaction between the user and the interface happens as follows:

1. the user first selects a *kernel* concept, i.e., the starting concept, from W1, which initially lists all domain concepts with their descriptions;
2. the selected concept appears on the graph (i.e., W3) as a *variable node* and becomes the *pivot/active/focus* node (i.e., the node coloured in orange or highlighted);
3. W2 displays the attributes of selected variable node in the form of text fields, range sliders, etc., so that the user can select them for output or constrain them;
4. the attributes selected for output (i.e., using the “eye" button) appear on the corresponding variable node in black with a letter “o", while constrained attributes appear in blue with letter “c";

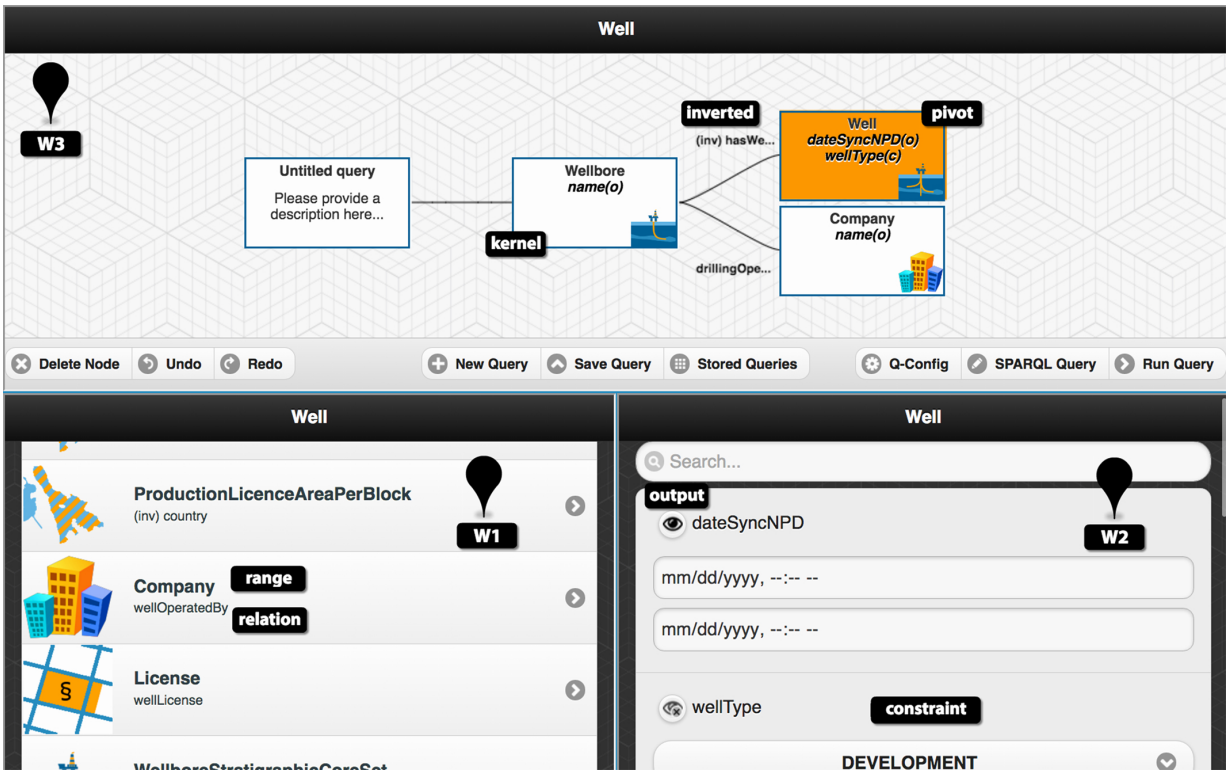

Fig. 2. OptiqueVQS interface – an example query in visual mode.

5. the user can further refine the type of variable node from W2, by selecting appropriate subclasses, which are treated as a special attribute (named “Type") and presented as a multi-selection combo-box form element;
6. once there is a pivot node, each item in W1 represents a combination of a possible relationship – range concept pair pertaining to the pivot (i.e., indeed a path of length one);
7. a selection of path/item in W1 triggers a join between the pivot and the new variable node (of type range concept) over the specified relationship, and the new variable node becomes the focus (i.e., *pivoting*).

The user has to follow the same steps to involve new concepts in the query and can always jump to a specific part of the query by clicking on the corresponding variable node in W3. The arcs that connect variable nodes do not have any direction, but it is implicitly left to right. This is because for each active node only outgoing relationships and inverses of incoming relationships are presented for selection in W1. An example query is depicted in Figure 2 for the Statoil use case. The query asks for the all the wellbores that belong to a development well and operated by a company. In the output, we want to see the name of wellbore, the synchronisation date and the name of the company.

The user can delete nodes, access the query catalogue, save/load queries, and undo/redo actions through affordances provided by the buttons at the bottom part of W3. W3 indeed acts as a master widget, since it possesses the whole query, and deals with its persistence. The user basically can re-use existing queries stored in the system by anyone, hence could modify an existing query to fit his/her current needs.

The user can also switch to editable SPARQL mode and see the textual form of a query by clicking on “SPARQL Query" button at the bottom-right part of the W3 as depicted in Figure 3. The user can keep interacting with the system in the textual form and continue to the formulation process by interacting with the widgets. For this purpose, pivot/focus variable node text is highlighted and every variable node text is associated with a hyperlink to allow users to change the focus. Availability of textual mode and its synchronisation with the visual mode enable us to realise col-

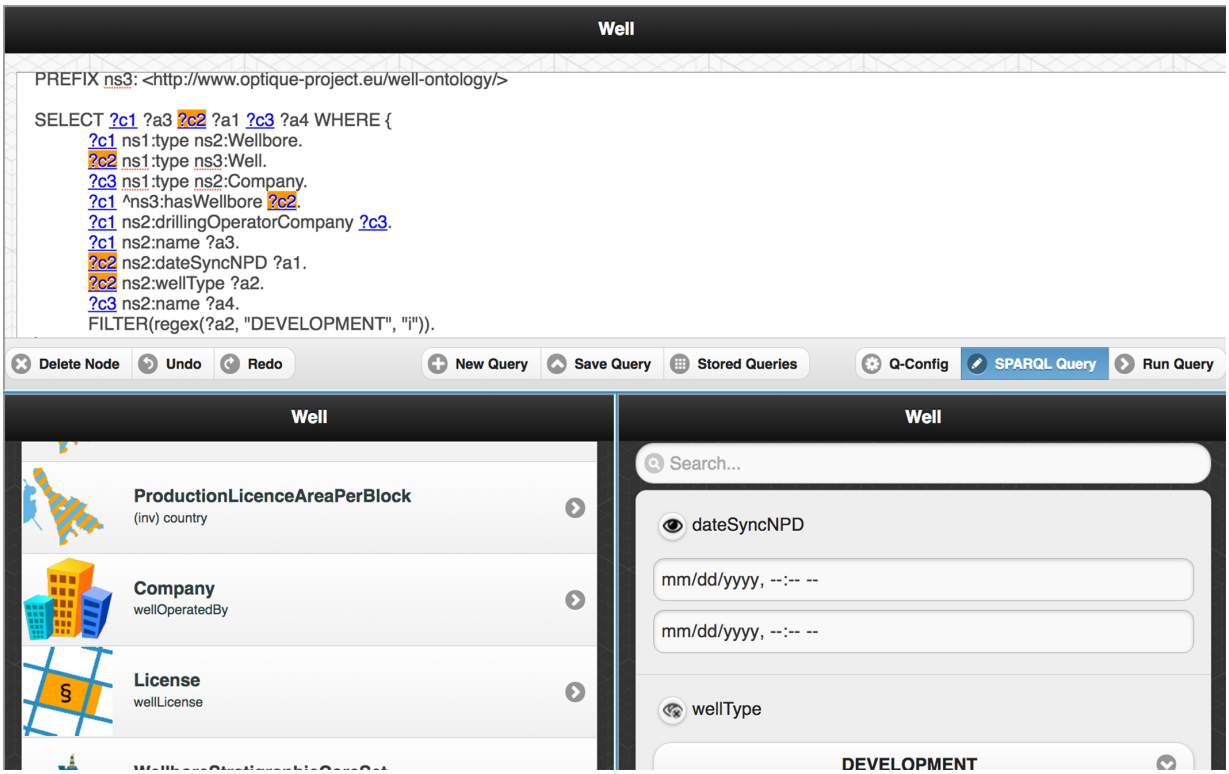

Fig. 3. OptiqueVQS interface – an example query in textual mode.

laboration between end users and IT experts. Particularly, for highly complex queries, IT experts could provide help on the textual mode, which they are expected to be more comfortable with, while end users could keep working on the visual mode. Moreover, from a didactic perspective, end users, who are eager to learn the textual query language, could switch between two modes and see the new query fragments being added/deleted after each interaction. Note that SPARQL mode is compliant, in terms of expressiveness, to what can be represented in the visual mode.

We extended OptiqueVQS with three new widgets, which provide an evidence on how a widget-based architecture allows us to hide complex functionality behind layers and combine different paradigms. One widget is for viewing example results and other two widgets are meant to address spatial and temporal use cases. They are activated by annotating (i.e., OWL annotations) relevant properties as temporal or spatial (recall R3 at Section 4). The widgets are described as follows:

(W4) The fourth widget is tabular result widget and appears as soon as the user clicks on the "Run Query' button (see Figure 4). It provides an example result list for the current query and also affordances for aggregation and sequencing operations.

Aggregation and sequencing operations fit naturally to a tabular view, since it is a related and familiar metaphor. Users can also view the full result list, inspect the individuals, and export data. For these purposes, in Optique, we use the Information Workbench (IWB) [38,51], which is a generic platform for semantic data management.

(W5) The fifth widget is a map widget. It is a domain-specific component for Statoil use case, and it allows end users to constrain attributes by selecting an input value from the map (see Figure 5).

A button with a pin icon is placed next to every appropriate attribute (i.e., annotated as spatial) presented in W2 to activate the map widget.

(W6) The third widget is a domain-specific component and supports temporal queries in the context of Siemens use case (see Figure 6).

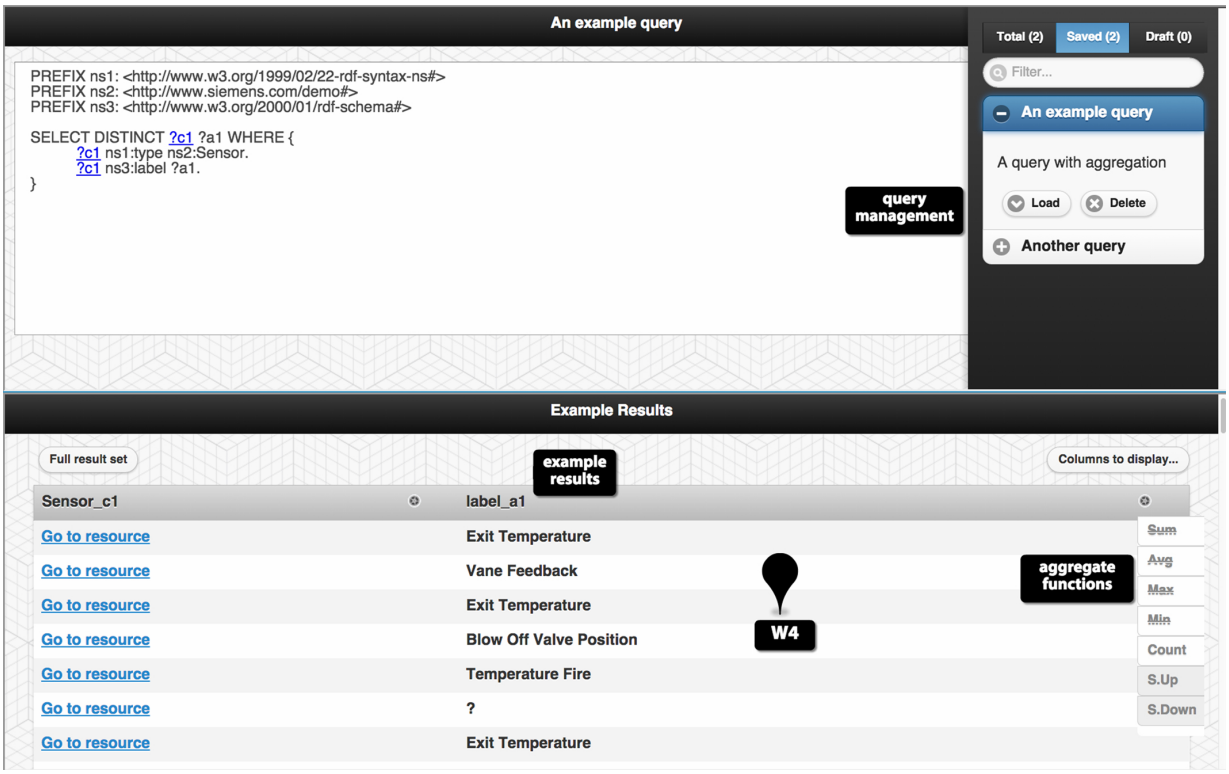

Fig. 4. OptiqueVQS interface – the tabular result widget with aggregation and sequencing support.

OptiqueVQS generates temporal queries in STARQL [62]. STARQL provides an expressive declarative interface to both historical and streaming data. OptiqueVQS switches to STARQL mode, when the user selects a dynamic property (i.e., whose extensions are time dependent, and coloured in blue). A stream button appears on top of W1 and lets the user configure parameters such as slide (i.e., frequency at which the window content is updated/moves forward) and window width interval. If the user clicks on the "Run Query" button, a template selection widget (W4) appears for selecting a template for each stream attribute, which is by default "echo" (see Figure 7); W4 is normally used for displaying example results in SPARQL mode. The example query depicted in Figure 6 and Figure 7 asks for a train with turbine named "Bearing Assembly", and queries for the journal bearing temperature reading in the generator. The user can register the query in W4 by clicking on the "Register query" button.

Finally, OptiqueVQS exploits the query history to rank and suggest ontology elements with respect to a *partial query* (in W1 and W2) that the user has constructed so far (i.e., context-aware) [73].

5.1.1. Design Rationale

The usability OptiqueVQS is built on several generic and local design choices. The former is addressed as a part of quality attributes in Section 6, while in this section we address the local design choices concerning the implementation of individual widgets. Major local design choices involve:

(a) *tree-shaped query representation* (W3) is meant to increase the comprehensibility compared to generic graph representations with arcs and nodes directed and placed to arbitrary points;
(b) *inverted object properties* (W1 and W3) ensure a direction-free query representation and navigation in order to decrease the cognitive load;
(c) *object property – range concept pairs* (W1) decrease the number of navigational steps; i.e., rather than selecting an object property then a range concept, the user can select a pair at a single step;
(d) *simplified type refinement* (W2) reduces the type refinement to the attribute level; that is, the list of

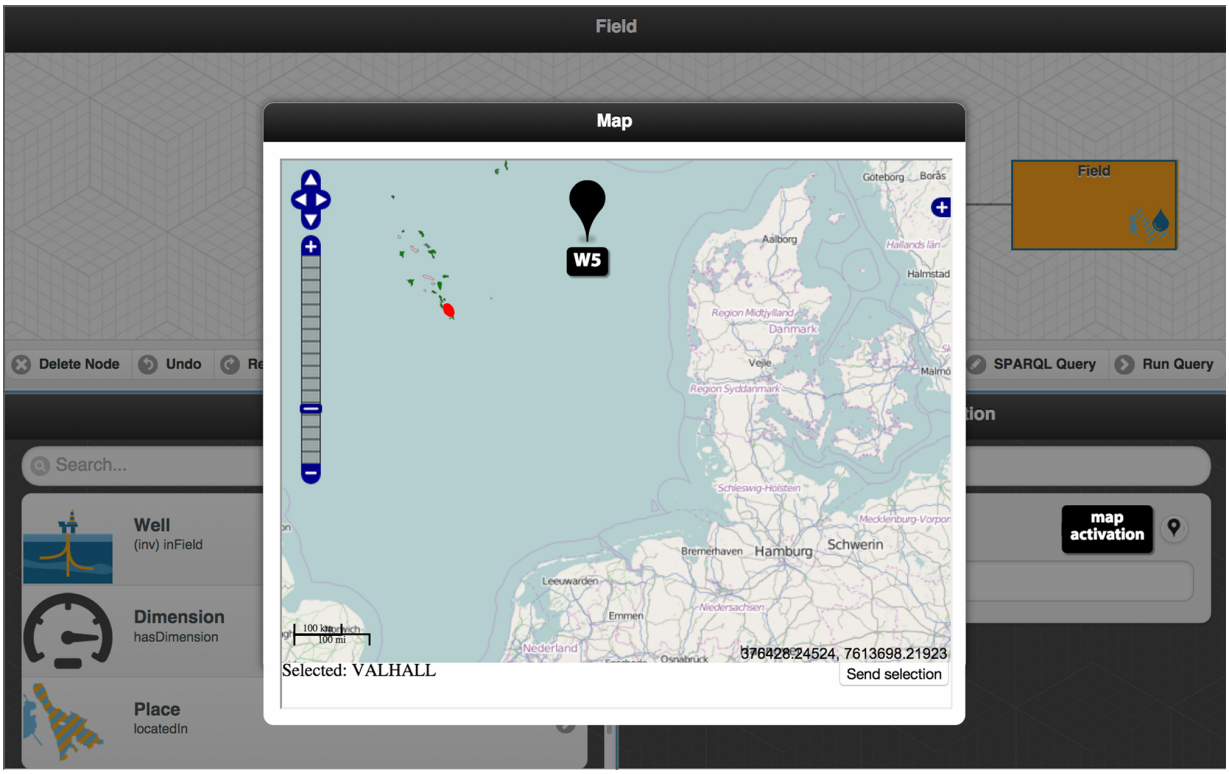

Fig. 5. OptiqueVQS interface – the map widget.

subclasses presented as an ordinary form element to provide a simplified solution.

In general, such design choices provide an orderly presentation and hides the jargon related to the graphs, query language, and ontologies, which end users should not worry about. The semantics and syntax of the underlying query language and ontology are not delivered as they are; however, a correct translation from end-user operations to the query language is ensured. The goal is to hide complexity and technical jargon effectively so as to reduce the knowledge and skills required.

For example, in a variant of OptiqueVQS, a graph representation is employed along with ingoing/outgoing arc distinction. In a user study with casual users, the participants complained about disorder in the presentation and their confusion due ingoing/outgoing relation distinction [90]. However, in another study with original OptiqueVQS with casual users, the participants praised the order and simplicity of tree-shaped presentation [74].

5.2. *OptiqueVQS backend*

In this section, we present the backend infrastructure OptiqueVQS relies on. In Figure 8, one can see the main components in OptiqueVQS backend.

The frontend communicates with the backend via a REST API that returns a JSON object according to the performed request. The backend is in charge of accessing *(i)* the ontology, which drives the information displayed in the frontend, and *(ii)* the query log, which plays an important role in ranking [73] as well as it serves as an example for the formulation of similar future queries.

The ontology can optionally be enriched with additional axioms to capture values that are frequently used and rarely changed (refer to "Data sampler" component in the architecture); this includes the list of values and numerical ranges in an OWL data property range (i.e., for max/min sliders and drop-down boxes in W2).

The main component of the backend is the "graph projector" (described in the next section), which will create a navigation graph according to the ontology axioms. The "graph projector" in conjunction with the

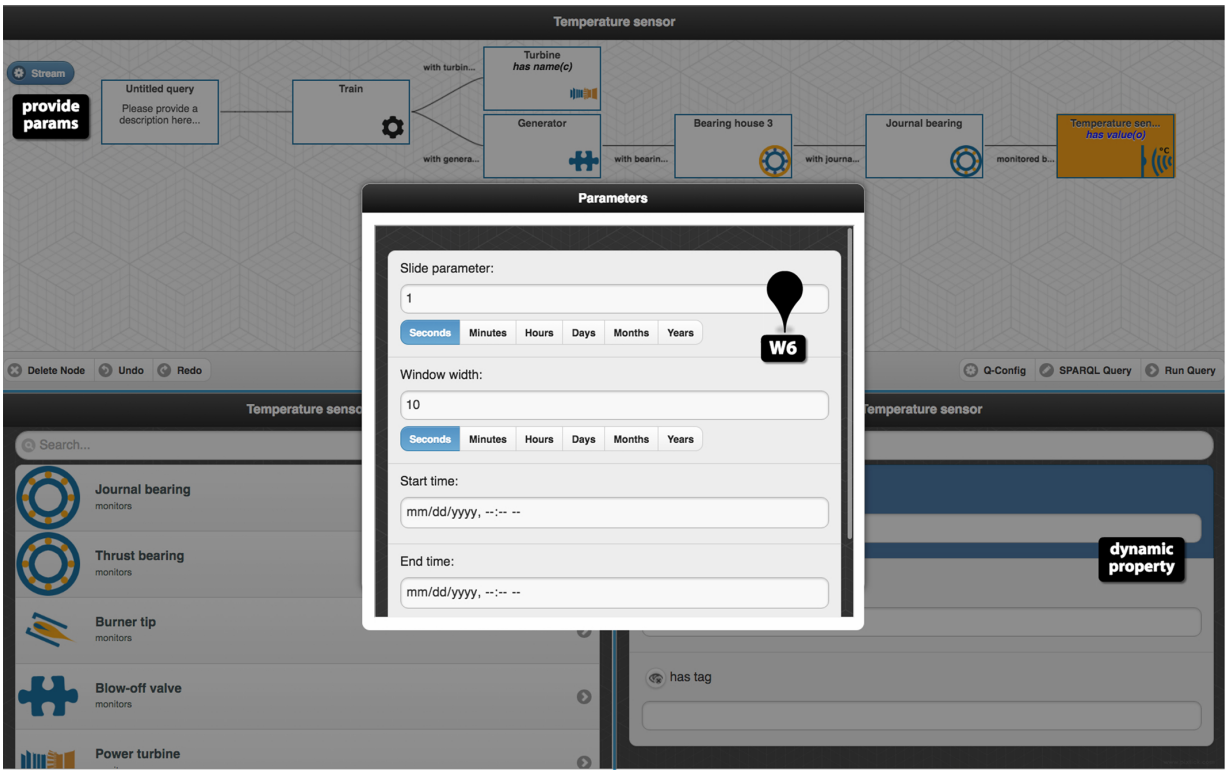

Fig. 6. OptiqueVQS interface – the stream parameter selection widget.

"VQS feeder" will drive the population of the frontend widgets.

5.2.1. *Ontology-driven Navigation Graph*

From our work on the use cases, we discovered that end users ask mostly schema-level queries, e.g., "give me all wellbores that are located in a certain area". Thus, we are targeting at query formulation that is done in terms of classes and properties. OWL 2 axioms, on the other hand, can be exploited to help a user in navigating between classes and properties. For example, if a user during query formulation has the concept Wellbore active, then a query formulation system could suggest them to connect Wellbore with Core via hasCore due to the axiom '*Wellbore* SubClassOf: *hasCore* some *Core*'. Moreover, most of users' queries have a graph-like structure, where nodes are labelled with concepts and edges with properties. However, OWL 2 axioms are not well-suited for a graph-based navigation. Indeed, note that OWL 2 axioms do not have a natural correspondence to a graph, e.g., an OWL 2 axiom of the form 'C_1 and C_2 SubClassOf: D_1 or D_2' can be hardly seen as a graph. Even in the case when an axiom can naturally be seen as a graph, to the best of our knowledge there is no standard means to translate it to a graph. Therefore, we need a technique to extract a suitable graph-like structure from a set of OWL 2 axioms. To this end, we have adapted a technique called *navigation graph* [8,7].

The nodes of a navigation graph are unary predicates, constants (named individuals, literal values) or datatypes, and edges are labelled with possible relations between such elements, that is, binary predicates. The key property of a navigation graph is that every X-labelled edge (v, w) is justified by one or more axioms entailed by $\mathcal{O}$ which "semantically relates" v to w via X.

Definition 5.1. *Let $\mathcal{O}$ be an OWL 2 ontology A* navigation graph *for $\mathcal{O}$ is a directed labelled multigraph G having as nodes unary predicates, constants or datatypes from $\mathcal{O}$ and s.t. each edge is labelled with a binary predicate from $\mathcal{O}$. Each edge e is justified by one or more axioms α_e s.t. $\mathcal{O} \models \alpha_e$ and α_e is of the form given next, where b is a named individual, l_i is a literal value, A, A_{sup}, A_{sub}, B classes or unary predicates, R_o, R_o^- object properties, R_d a datatype property,*

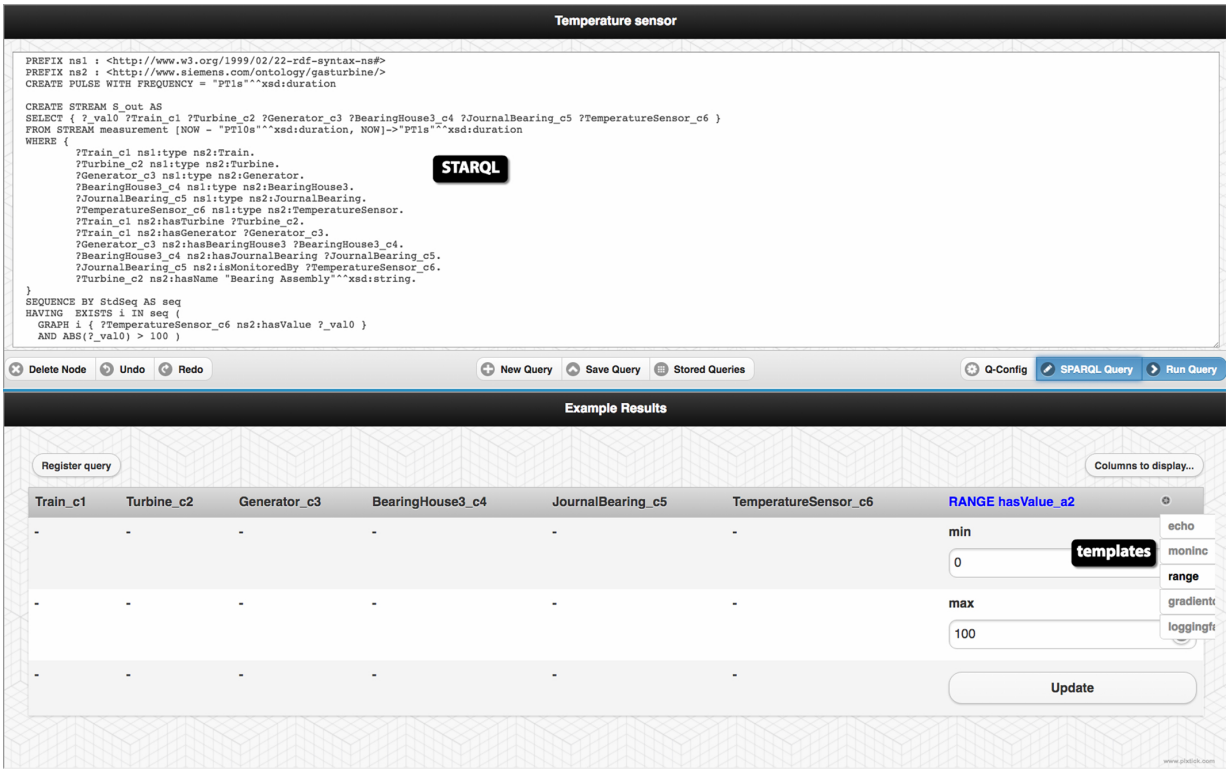

Fig. 7. OptiqueVQS interface – template selection for a stream query.

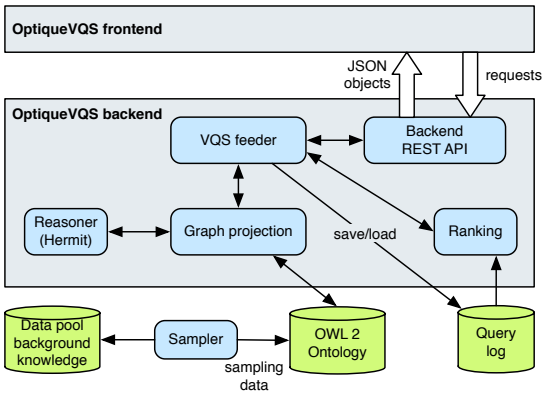

Fig. 8. OptiqueVQS backend.

dt a datatype (e.g. string, integer), and x, y numerical values:

(i) Edges e of the form $A \xrightarrow{R_o} B$ are justified by the following OWL 2 axioms:

– 'A SubClassOf: R_o restriction B', where *restriction* is one of the following: *some* (existential restriction), *only* (universal restriction), *min x* (minimum cardinality), *max x* (maximum cardinality) and *exactly x* (exact cardinality). Note that axioms of the form 'A SubClassOf: R restriction $\bigsqcup_{1\leq i\leq n} B_i$' and '$A$ SubClassOf: R restriction $\bigsqcap_{1\leq i\leq n} B_i$' also justify edges of the form $A \xrightarrow{R_o} B_i$.

– A combination of range and domain axioms of the form: R_o Domain: A' and 'R_o Range: B'.

– 'A SubClassOf: R_o value b', and b being a member of the class B (e.g., 'b Types: B').

– 'R_o InverseOf: R_o^-' when the navigation graph includes the edge $B \xrightarrow{R_o^-} A$.

– Top-down propagation of restrictions: 'A SubClassOf: A_{sup}' when the navigation graph includes the edge $A_{sup} \xrightarrow{R_o} B$.

– Bottom-up propagation of restrictions: 'A_{sub} SubClassOf: A' when the navigation graph includes the edge $A_{sub} \xrightarrow{R_o} B$.

(ii) Edges e of the form $A \xrightarrow{R_d} dt$ are justified by the following OWL 2 axioms:

- 'A SubClassOf: R_d restriction dt', where *restriction* is one of the following: *some*, *only*, *min x*, *max x* and *exactly x*. Note that dt can be a OWL 2 built-in datatype or user-defined datatype which are typically expressed with a datatype restriction (e.g., 'A SubClassOf: R_d restriction $dt[> x, < y]$', where dt is restricted with the interval defined by x and y.)
- A combination of range and domain axioms of the form: R_d Domain: A' and 'R_d Range: dt' (or 'R_d Range: $dt[> x, < y]$').
- 'A SubClassOf: R_d value l', and l being a literal value of type dt.
- Top-down propagation of restrictions: 'A SubClassOf: A_{sup}' when the navigation graph includes the edge $A_{sup} \xrightarrow{R_d} dt$.
- Bottom-up propagation of restrictions: 'A_{sub} SubClassOf: A' when the navigation graph includes the edge $A_{sub} \xrightarrow{R_d} B$.

(iii) Edges e of the form $A \xrightarrow{R_d} l_i$ are justified by the following OWL 2 axioms:

- A SubClassOf: R_d restriction $\{l_1 \dots l_n\}$, where *restriction* is one of the following: *some*, *only*, *min x*, *max x* and *exactly x*; and $l_1 \dots l_n$ is an enumeration of literal values (typically of type 'string').
- A combination of range and domain axioms of the form: R_d Domain: A' and 'R_d Range: $l_1 \dots l_n\}$'.
- 'A SubClassOf: R_d value l_i'.

(iv) Edges e of the form $A \xrightarrow{broader} B$ are justified by the OWL 2 axiom: B SubClassOf: A.

The edges in the navigation graph are used to populate the frontend widgets with suggestions to guide the end user in the formulation of the query. Edges of type (i) are used to populate W1, while edges of types (ii) and (iii) populate the attributes in W2, for the focus concept A in W3. Edges of type (iii) and (ii) also guide the automatic customization of W2 with specific input fields for a given datatype, pre-populated drop-down lists for enumeration of values (e.g., company names) and range sliders for datatype restrictions (e.g., min/max possible depth of wellbores). Edges of type (iv) populate the list of subclasses for the focus concept A, which are treated as the special attribute "Type" in W2. OptiqueVQS relies on the OWL 2 reasoner HermiT [32] to build the navigation graph (e.g., extraction of classification) in order to consider both explicit and implicit knowledge defined in the ontology O.

The number of suggestions presented in W1 and W2 may grow quickly due to ontology size, number of relationships between concepts, inverse properties, and the propagative effect of inheritance of restrictions etc. As the lists grow, the time required for a user to find elements of interest increases; therefore, *adaptive query formulation*, that is ranking ontology elements with respect to previously executed queries (i.e. a query log), is a critical aspect in OptiqueVQS. OptiqueVQS implements a light version of the ranking method described by Soylu et al. [73].

5.2.2. Query Conformation to Navigation Graph

To realise the idea of ontology and data guided navigation, we require that interfaces *conform to* the navigation graph in the sense that the presence of every element on the interface is supported by a graph edge. In this way, we ensure that interfaces mimic the structure of (and implicit information in) the ontology and data and that the interface does not contain irrelevant (combinations of) elements.

Our goal is to help a user to construct such queries that would be "justified" by the navigation graph. We assume that all the definitions in this section are parametrised with a fixed ontology $\mathcal{O}$.

Definition 5.2. *Let Q be a conjunctive query. The* graph of Q *is the smallest multi-labelled directed graph G_Q with a node for each term in Q and a directed edge (x, y) for each atom $R(x, y)$ occurring in Q, where R is different from $\approx$. We say that Q is* tree-shaped *if G_Q is a tree. Moreover, a variable node x is labelled with a unary predicate A if the atom $A(x)$ occurs in Q, and an edge (t_1, t_2) is labelled with a binary predicate R if the atom $R(t_1, t_2)$ occurs in Q.*

Finally, we are ready to define the notion of conformation.

Definition 5.3. *Let Q be a conjunctive query and G a navigation graph. We say that Q* conforms to *G if for each edge (t_1, t_2) in the graph G_Q of Q the following holds:*

- *If t_1 and t_2 are variables, then for each label B of t_2 there is a label A of t_1 and a label R of (t_1, t_2) such that $A \xrightarrow{R} B$ is an edge in G.*
- *If t_1 is a variable and t_2 is a constant, then there is a label A of t_1 and a label R of (t_1, t_2) such that $A \xrightarrow{R} t_1$ is an edge in G.*

Now we describe the class of queries that can be generated using OptiqueVQS and show that they conform to the navigation graph underlying the system. First, observe that the OptiqueVQS queries follow the following grammar:

$$\begin{aligned} \texttt{query} &::= A(x)(\wedge\ \texttt{constr}(x))^*(\wedge\ \texttt{expr}(x))^* \\ \texttt{expr}(x) &::= \texttt{sug}(x,y)(\wedge\ \texttt{constr}(x))^*(\wedge\ \texttt{expr}(y))^* \\ \texttt{constr}(x) &::= \exists y R(x,y) \mid R(x,y) \mid R(x,c) \\ \texttt{sug}(x,y) &::= Q(x,y) \wedge A(y) \end{aligned}$$

where A is an atomic class, R is an atomic data property, Q is an object property, and c is a data value. The expression of the form $A(\wedge\ B)^*$ designates that B-expressions can appear in the formula 0, 1, and so on, times. An OptiqueVQS `query` is constructed using suggestions `sug` and constraints `constr`, that are combined in expressions `expr`. Such queries are clearly conjunctive and tree-shaped (recall R1 at Section 4). All the variables that occur in classes and object properties are output variables and some variables occurring in data properties can also be output variables.

When users interacts with OptiqueVQS,

- They start with a "starting" class, as described above. Clearly, this initial query conforms to any navigation graph, including the one, underlying the system.
- Then, the system suggest the list of $\texttt{sug}(y,z)$ via W1 and of $\texttt{constr}(x)$ via W2 such that choosing any of them would leave the updated query conforming to the underlying navigation graph. In other words, all these choices are justified by the graph.

6. Quality Features

OptiqueVQS has the following interrelated features that are mapped to the quality attributes (i.e., A*n*) proposed at Section 4.2:

(F1) *View and overview* provide a continuous outlook of the query formulated so far while supplying the user with a set of possible actions. The goal is to ensure maximum end-user awareness and control (cf. [71]) (A1).
Realisation: W3 provides a global overview of the user query, while W1 and W2 focus the user to the pivot for possible join, select, and projection operations.

(F2) *Exploration and construction* allow the user to navigate the conceptual space for exploration and construction purposes. Exploration could be also at instance level, in terms of cues (i.e., sample results) and instance level browsing (cf. [22,66]) (A1).
Realisation: W1 and W2 suggest domain elements and allow ontology navigation. Each action adds reversible query fragments into the query. The user can also use the tabular result widget, i.e., W4, for example results.

(F3) *Collaborative query formulation* is meant to enable collaboration between users actively or passively. Such collaboration could be between an end user and an IT expert or between an end users (cf. [59]) (A1). Users can formulate more complex queries and improve their effectiveness and efficiency.
Realisation: OptiqueVQS synchronises visual and textual modes (i.e., active collaboration between IT experts and end users), allows users to share queries (i.e., passive collaboration), and harnesses the query log to offer suggestions (i.e., passive).

(F4) *Query-reuse* enables the user to reuse existing queries as they are or to modify them to construct more complex queries and/or to improve the effectiveness and efficiency (A1 and A9). Query reuse could indeed be considered a passive form of collaboration (cf. [59]) (F3).
Realisation: OptiqueVQS allows users to store, load, and modify queries. Queries are stored in a query catalogue with descriptive texts to facilitate their search and retrieval.

(F5) *Spiral/layered design* refers to distributing system functionality into layers (cf. [67]), so as to enable an orderly access to system, prevent complex functionalities to hinder the usability for less competent users (A1), view ontology at different levels of detail (A3), tailor available functionality with respect to user needs (A4 and A5), and to add new functionalities without overloading the interface (A6).
Realisation: OptiqueVQS delegates functionality and ontology visualisation tasks to the different widgets. For instance, W4 offers aggregation and sequencing operations, while W2 presents data attributes and offers selection and projection functions.

(F6) *Gradual access* is to cope with large ontologies with many concepts and properties. The amount

of information that can be communicated on a finite display is limited. Therefore, gradual and on-demand access to the relevant parts of an ontology is necessary (cf. [45]) (A1 and A3).
Realisation: W1 and W2 provide ontology elements adaptively and gradually on user demand, hence avoids cluttering and scattering the interface.

(F7) *Iterative formulation* allows the user to follow a formulate-inspect-reformulate cycle (A1), since a query is often not formulated in one iteration (cf. [87,59]).
Realisation: OptiqueVQS provides affordances to inspect, manipulate and extend a formulated query. For instance, users can freely change the pivot, delete nodes, and add new nodes from any point of the query.

(F8) *Ranked suggestions* improve the user efficiency by ranking ontology elements with respect to context, e.g., previous query log, and filtering down the amount of knowledge to be presented (cf. [73]) (A1, A3, and A4). Ranking is a form of passive collaboration as it utilises queries formulated by others to provide gradual access (F3 and F6).
Realisation: OptiqueVQS offers a ranking method, which exploits the query history of users to rank and suggest ontology elements (in W1 and W2) with respect to a partial query that a user has constructed so far.

(F9) *Domain specific representations* support varied data types and domains. This ensures contextual delivery of data leading to immediate grasping (cf. [85]) (A1). The availability of domain specific representations provides users and system with the opportunity to select representation paradigms that fit best to the data and task (A4 and A5).
Realisation: OptiqueVQS allows introducing new domain-specific widgets for visualisation and interaction, for instance, the map widget (W5) for geospatial interaction and visualisation.

(F10) *Multi-paradigm and multi-perspective* presentation is meant to combine multiple representation and interaction paradigms, such as form and diagrams, and query formulation approaches, such as visual query formulation and textual query editing, to meet diverse contexts. (cf. [22,45]) (A1). Moreover, the system and users can adapt presentation (A4 and A5) and users can select among various paradigms depending on their role (F3), task (F2), and data at hand (F9).
Realisation: OptiqueVQS puts multiple representation and interaction paradigms (i.e., list/menus (W1), diagrams (W3), forms (W2), tables (W4)) as well as query formulation approaches (i.e., textual and visual) together.

(F11) *Modular architecture* allows new components to be easily introduced and combined in order to adapt to changing requirements and to support diverse user experiences (A1, A2 and A6). This could include alternative/complementary components for query formulation, exploration, visualisation, etc. with respect to context (A3, A4, A5, F9, and F10).
Realisation: OptiqueVQS is based on a UI mashup approach and is built on a widget-based architecture, where widgets are independent components acting as the building blocks. They communicate through broadcasting event notifications.

(F12) *Data exporting* enables the user to feed analytics tools with the data extracted for sense-making processes, as they are not expected to have skills to transform data from one format to another. Therefore, means to export data in different format are required to ensure that the system fits into the organisational context (A7) and a broader user experience (A1).
Realisation: OptiqueVQS allows users to export data in various formats. For instance, in the context of Statoil use case, users can export query results in the format of their data analytics tools.

(F13) *Domain-agnostic backend* ensures domain independence. This allows VQS to operate over different ontologies and datasets without any extensive manual customisation and code change [46,57] (A8).
Realisation: OptiqueVQS relies on a domain-agnostic backend. It projects the underlying ontology into a graph for exploration and query construction. Yet, it also allows domain-specific components to be introduced.

7. User Evaluation

The purpose of a VQSs is to enable users to formulate queries effectively and efficiently. The *effectiveness* (cf. [21,14]) is measured in terms of accuracy and completeness that users can achieve. The cost associated with the level of effectiveness achieved is

called *efficiency* (cf. [21,14]), and is mostly measured in terms of the time spent to complete a query. Note that, typically in *information retrieval* (IR), effectiveness is measured in terms of *precision*, *recall*, and *f-measure* (harmonic mean of precision and recall) over the result set; however, a VQS is a *data retrieval* (DR) paradigm, for which a single missing or irrelevant object implies a total failure [72]. In other words, data retrieval systems have no tolerance for missing or irrelevant results, while IR systems are variably insensitive to inaccuracies and errors, since they often interpret the original user query and the matching is assumed to indicate the likelihood of the relevance, rather than being exact (cf. [88,10]). Therefore, for a VQS, effectiveness is rather measured in terms of a binary measure of success (i.e., correct/incorrect query) (cf. [46]).

In the course of Optique project, we conducted a total of four industrial workshops with our use case partners (two at each use case). In the first set of workshops, we conducted unstructured interviews with domain experts and observed them in their daily routines. Shortly after the first set of workshops, we demonstrated a paper mock-up and had further discussions. A running prototype was developed iteratively with representative domain experts in the loop. At the second round of workshops, domain experts experimented with the prototype in a formal think aloud session and we measured the effectiveness and efficiency of OptiqueVQS.

In parallel, we conducted two usability studies with casual users to have rapid feedback, which are published elsewhere, in a non-industrial context as the availability of domain experts is often limited:

(Exp1) *An experiment involving casual users* without any technical skills and knowledge. It was conducted on a generic domain. The results suggested that casual users without any technical background can effectively and efficiently use OptiqueVQS to formulate complex queries [74].

(Exp2) *A comparative experiment* comparing a variant of OptiqueVQS and a form-based query interface called PepeSearch [89]. The results suggested that OptiqueVQS is the preferred tool for formulating complex query tasks, where PepeSearch is the preferred tool for less experienced users for completing simple tasks [90].

In this article, we report the design and the results of the experiments that we conducted with our industrial partners:

(Exp3) *Statoil experiment* employed on a bootstrapped (i.e., automatically generated [68,50]) oil and gas ontology[5] with 253 concepts, 208 relationships (including inverse properties), and 233 attributes [77].

(Exp4) *Siemens experiment without temporal queries* employed a manually constructed diagnostic ontology with five concepts, five relationships (excluding inverse properties), and nine attributes [53].

(Exp5) *Siemens experiment with temporal queries* employed a manually constructed turbine ontology with 40 concepts and 65 properties [76].

The ontologies, data, and information needs used in the experiments are provided by the industrial partners themselves and therefore are not artificial and reflect the reality and real interests.

7.1. Experiment design

The experiments were designed as a *think-aloud* study. Each participant performed the experiment in a single session, while being watched by an observer. Participants were instructed to think aloud, including any difficulties they encounter (e.g., frustration and confusion), while performing the given tasks. A five minutes introduction of the topic and tool had been delivered to the participants along with an example before they were asked to fill in a profile survey. The survey asked users about their age, occupation and level of education, and asked them to rate their technical skills, such as on programming and query languages, and their familiarity with similar tools on a Likert scale (i.e., 1 for "not familiar at all," 5 for "very familiar"). Participants were then asked to formulate a set of information needs into queries with OptiqueVQS (i.e., tasks).

A number of empty queries, each corresponding to a task in the experiment, was generated in OptiqueVQS for each user. Users received their tasks one by one on paper, and for each task loaded the corresponding empty query. Formulating and executing a query, i.e., clicking "run query" button, and inspecting the result set equals to one attempt. Participants had a maximum of three attempts per task and this was enforced by the system ("run query" button was blocked after three attempts). A task was ended, when the participant acknowledged the completion or exhausted his/her three

[5] http://sws.ifi.uio.no/project/npd-v2/

Table 3
Profile information of the participants.

#	Age	Occupation	Exp.	Education	Tech. skills	Similar tools	Sem. Web
P1	39	Geologist	Exp3	Master	3	3	1
P2	40	Biostrat	Exp3	Master	2	1	1
P3	49	IT advisor	Exp3	Master	5	4	1
P4	33	Software engineer	Exp4	Bachelor	5	2	1
P5	27	Diagnostic Engineer	Exp4	Bachelor	5	5	1
P6	60	Mechanical Engineer	Exp4	Master	3	1	1
P7	45	Mechanical Engineer	Exp4	Bachelor	1	2	1
P8	37	R&D engineer	Exp5	PhD	4	1	1
P9	54	Diagnostics Engineer	Exp5	Bachelor	5	3	1
P10	39	Engineer	Exp5	PhD	5	2	1

attempts. Every attempt for each task was recorded by the OptiqueVQS as a draft query, along with the time it had taken for each attempt.

Three participants from Statoil and seven participants from Siemens took part in the experiments. The profiles of participants are summarised in Table 3, which shows that participants vary in technical skills and experience with similar tools and have no familiarity with the semantic web technologies.

There were nine tasks for the Statoil experiment (Exp3), five tasks for the Siemens experiment (Exp4), and five tasks for the second Siemens experiment with temporal queries (Exp5). The tasks were all conjunctive and shown in Table 4. The key elements are highlighted in the context of this article for clarity.

7.2. Results

The results of all the three experiments are summarised in Figure 9.

Regarding the Statoil experiment (Exp3), a total of 27 tasks was completed by the participants, with 84 percent correct completion rate and 69 percent first-attempt correct completion rate (i.e., percentage of correctly formulated queries in the first attempt). The first participant had only one incorrect, and the second participant had no incorrect task. T3 was about fields operated by Statoil, and the third participant formulated a Field - FieldOperator pair instead of a Field - Company pair. This confusion between FieldOperator and Company led him to incorrectly solve the T5 as well. T7 not only takes the longest time but also the highest average attempts. According to participant feedback and our observations, this is particularly due to conceptual mismatch between the participants'understanding of domain and the ontology (the ontology was automatically bootstrapped with little manual fine tuning), which forced participants to iterate several times.

In the Siemens experiment without temporal queries (Exp4), a total of 18 tasks was completed by the participants. The third participant exceeded the allocated time for the session and could not attempt the last two tasks, therefore these are omitted from the results. Correct completion rate was 88 percent and first-attempt correct completion rate was 72 percent. The third and fourth participants had one incorrect task. The Siemens diagnostic ontology used in the experiment was smaller in size and was manually created (i.e., of higher quality). Participants had only a minor issue with the date format, therefore Task 11, where a date constraint appeared for the first time, took the longest time.

In the Siemens experiment with temporal queries (Exp5), a total of 15 tasks were completed by the participants with 100 percent correct completion rate and 66 percent first-attempt correct completion rate. They had minor issues with the fact that users need click on the “Run Query” button in order to select a template from the tabular view. A straight forward solution for stream based queries would be to change the name of button to “Select a Template” to prevent confusion, as the “Run Query” button is originally meant for non-stream query tasks.

In general, participants raised two major issues. First, they asked for a longer training session; indeed, the training sessions were intentionally kept short in order to test learnability of the OptiqueVQS. The high completion rates, even with complex queries, suggest that the tool has high learnability. Secondly, participants pointed that the ontology did not always reflect their understanding of the domain, which was mostly a issue for the Statoil experiment. We acknowledge the situation and believe that the usability of an ontology

Table 4
Information needs used in the experiments – marked queries (*) are temporal.

#	Exp.	Information need
T1	Exp3	List all *fields*.
T2	Exp3	What is the water depth of the “Snorre A" *platform* (facility)?
T3	Exp3	List all *fields* operated by “Statoil Petroleum AS" *company*.
T4	Exp3	List all *exploration wellbores* with the *field* they belong to and the *geochronological era(s)* with which they are recorded.
T5	Exp3	List the *fields* that are currently operated by the *company* that operates the “Alta" *field*.
T6	Exp3	List the *companies* that are *licensees* in *production licenses* that own *fields* with a re-coverable oil equivalent over more than “300" in the *field reserve*.
T7	Exp3	List all *production licenses* that have a *field* with a *wellbore* completed between “1970" and “1980" and recoverable oil equivalent greater than “100" in the *company reserve*.
T8	Exp3	List the *blocks* that contain *wellbores* that are drilled by a *company* that is a *field operator*.
T9	Exp3	List all *producing fields* operated by “Statoil Petroleum AS" *company* that has a *wellbore* containing “gas" and a *wellbore* containing “oil".
T10	Exp4	Find all *assemblies* that exist in system.
T11	Exp4	Show all *messages* that *tribune* “NA0101/01" generated from “01.12.2009" to “02.12.2009".
T12	Exp4	Show all *turbines* that sent a *message* containing the text “Trip” between “01.12.2009" and “02.12.2009".
T13	Exp4	Show all *event* categories known to the system.
T14	Exp4	Show all *turbines* that sent a *message* category “Shutdown” between “01.12.2009" and “02.12.2009".
T15	Exp5	Display all *trains* that have a *turbine* and a *generator*.
T16	Exp5	Display all *turbines* together with the *temperature sensors* in their *burner tips*. Be sure to include the *turbine name* and the *burner tags*.
T17*	Exp5	For the *turbine named* “Bearing Assembly”, query for *temperature readings* of the *journal bearing* in the *compressor*. Display the reading as a simple echo.
T18*	Exp5	For a *train* with *turbine* named “Bearing Assembly”, query for the *journal bearing temperature* reading in the *generator*. Display readings as a simple echo.
T19*	Exp5	For the *turbine* named “Burner Assembly”, query for all *burner tip temperatures*. Display the readings if they increase monotonically.

is as crucial as the usability of a query formulation tool. Ontology usability is an overlooked issue in the research community, which demands more attention.

Overall, the results indicate high effectiveness and efficiency rates. This suggests that OptiqueVQS is a viable tool to visually construct considerably complex queries for querying structured data sources. All participants praised the capability of OptiqueVQS for formulating complex information needs into queries. A common statement was that such a solution will not only improve their current practices, but also augment their value creation potential due to the flexibility of formulating arbitrary queries. Three complex queries formulated by Statoil and Siemens domain experts and casual users are given in Figure 10. First query was provided by a Statoil domain expert for the query catalogue and he estimated that he would need a full day to extract this information with existing tools. On the other hand, the same Statoil user was able to formulate a query of a similar complexity with OptiqueVQS within less than 10 minutes. The second query (Exp3) only took 63 on average to complete by Siemens' domain experts. The third query took only 91 seconds to complete on average for a casual user [74].

8. Related Work

Visual approaches for querying structured semantic data sources are primarily categorised into VQLs and VQSs as explained in Section 1. Such approaches could be further classified with respect to main interaction paradigm.

Browsing and schema navigation are two prominent interaction paradigms. The former refers to interacting at an instance level, that is, the user browses the data set by adding and removing constraints and following the links between instances. Faceted search is a very good example of this paradigm, e.g., [86,16,8]. The latter is used by OptiqueVQS and refers to interacting at a conceptual level, that is using an external vocabulary, for example provided by an ontology, to express the information need at the schema level, e.g., [5,36]. Browsing is a good approach when the data set and re-

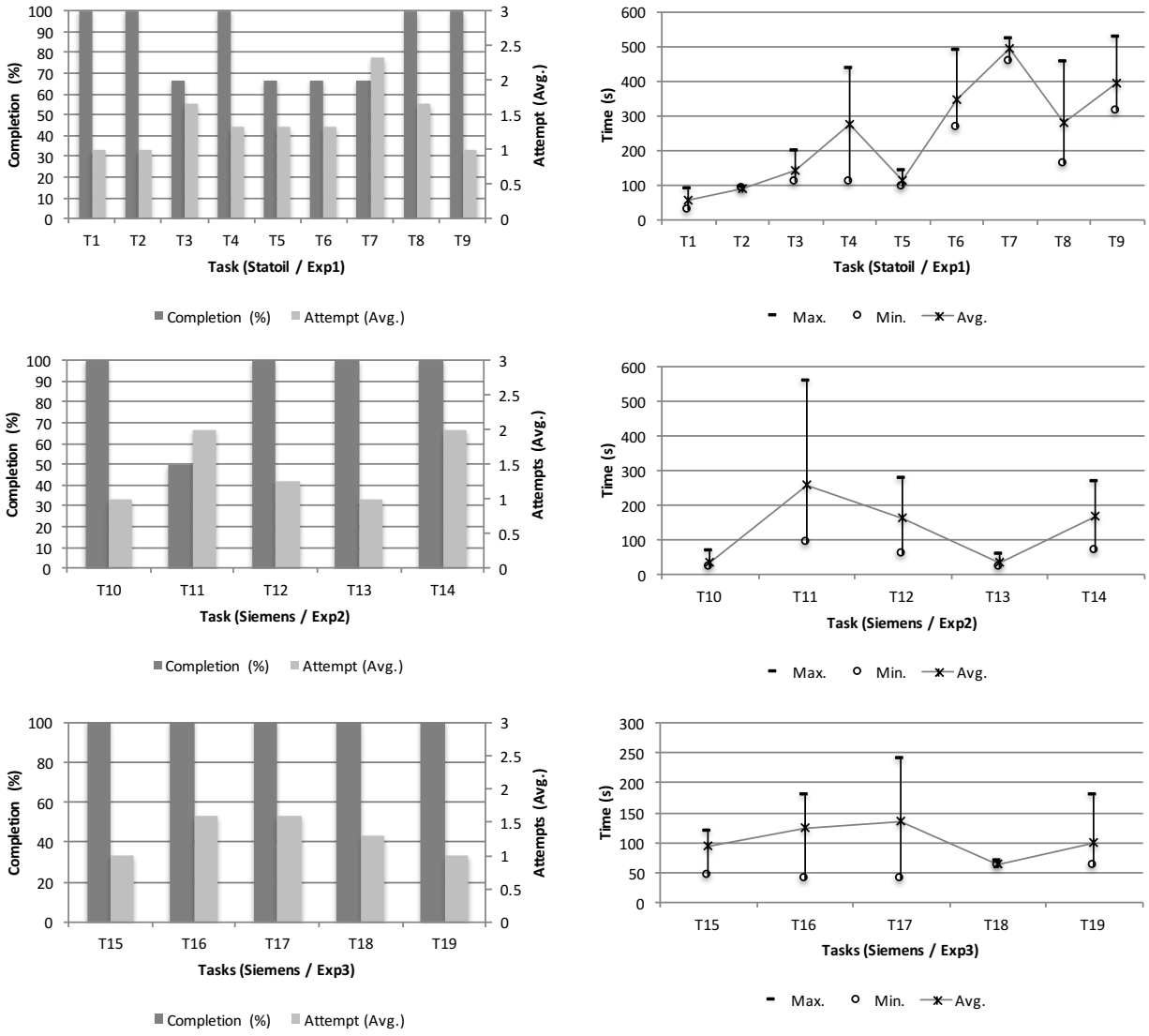

Fig. 9. Experiment results for the three usability studies.

sult set are not very large and users need to pay attention to each individual item in the result set. On the other hand, schema navigation works better when both data and results sets are large and users are interested in the result set itself and its correctness and completeness; this actually describes the situation in our industrial use cases.

Another categorisation arises from the source of vocabulary, which might be extracted from the data set or be provided by an external ontology. The former refers to extracting concepts and relationships by analysing the data set, i.e., extracting a pseudo ontology, e.g., [16,89]. It is adequate for cases where an ontology is not available, prevents the user from building unsatisfiable queries (i.e., no empty result sets), and allows using statistics about data for optimisation. The latter approach uses an ontology to feed the query formulation process, e.g., [82]. An ontology could be much more expressive than what one can extract from data, and vocabulary extraction process could be quite expensive for large and dynamic data sets. For example, data sets in our use cases change very rapidly in vast amounts and this makes real time processing very hard. Offline processing is not an option as this would lead to missing and/or incorrect results; users need to access real time data. Finally, it is not always desirable to formu-

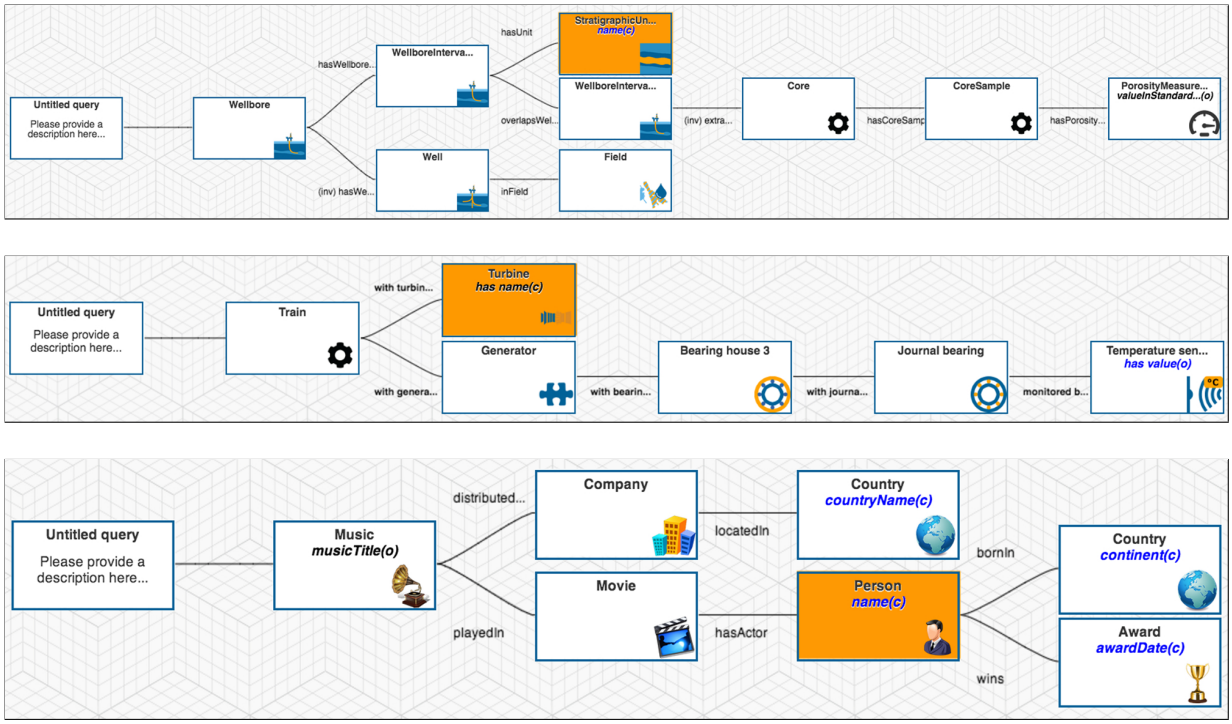

Fig. 10. Three complex queries formulated by Statoil and Siemens domain experts and casual users.

late queries with guaranteed results. For example, in the Siemens use case, most of the user queries specify an error situation in their hardware for which there is often no matching data at the time of query formulation (i.e., only to get notified when the data changes and the query becomes satisfied). OptiqueVQS uses a hybrid approach, where an ontology is the main source of vocabulary and data set is used for a limited extent (recall Section 5.2).

Regarding the visual approaches in general, notable examples of VQLs are LUPOSDATE [35], RDF-GL [42], Nitelight [69], GQL [12], and QueryVOWL [37]. LUPOSDATE, RDF-GL and Nitelight follow RDF syntax at a very low level through node-link diagrams representing the subject-predicate-object notation, while GQL and QueryVOWL represent queries at comparatively higher level, such as with UML-based diagrams. Each of these languages are managed by a VQS providing means for construction and manipulation of queries in a visual form. Albeit VQL-based approaches with higher level of abstraction are closer to end users, they still need to posses a higher level of knowledge and skills to understand the semantics of visual notation and syntax and to use it. Note that although OptiqueVQS uses a three-shaped query representation, it is informal, simplified, and free of any syntax and jargon related to ontologies and query languages.

A VQS have a better potential to offer a good balance between expressiveness and usability. The prominent examples of VQSs are gFacet [40], OZONE [82], SparqlFilterFlow [36], Konduit VQB [5], and Rhizomer [16], PepeSearch [89]. gFacet, OZONE, and SparqlFilterFlow employ a diagram-based approach and diagrams representing the queries are rather informal. Konduit VQB and Rhizomer employ a form-based paradigm. Diagram-based approaches are good in providing a global overview; however, they remain insufficient alone for view (i.e., zooming into a specific concept for filtering and projection). This is because the visual space as a whole is mostly occupied for query overview. Form-based approaches provide a good view; however, they provide a poor overview, since the visual space as a whole is mostly occupied with the properties of the focus concept. Approaches combining multiple representation and interaction paradigms are known to be better as they could combine view and overview. gFacet and Rhizomer are originally meant for data browsing, that is they operate on data level rather than schema level and every user

Table 5

Comparison of related tools with respect to our industrial requirements (B = Browsing, S = Schema navigation, H = Hybrid, O = Ontology, D = Data; ✓= yes, Θ = partially, - = no).

Criteria/Tool	gFacet	OZONE	SparqlFilterFlow	Konduit VQB	Rhizomer	PepeSearch	Super Stream	TELIOS Spatial	OptiqueVQS
Interaction type	B	S	S	S	B	H	S	S	S
Vocabulary	D	O	D	D	D	D	D	O	H
Downloadable	✓	-	-	-	✓	✓	-	-	✓
Three-shaped	✓	✓	✓	✓	✓	Θ	✓	✓	✓
Multi-paradigm	Θ	Θ	Θ	-	✓	-	-	✓	✓
Temporal	-	-	-	-	-	-	✓	-	Θ
Spatial	-	-	-	-	-	-	-	✓	Θ
Modularity	-	✓	-	-	-	-	✓	-	✓
Scalability	Θ	Θ	Θ	Θ	Θ	Θ	Θ	Θ	✓
Adaptivity	-	-	-	-	-	Θ	-	-	✓
Adaptability	-	✓	-	-	-	-	✓	-	✓
Extensibility	-	✓	-	-	-	-	✓	-	✓
Interoperability	-	-	-	-	-	-	-	Θ	Θ
Portability	✓	✓	✓	✓	✓	✓	✓	✓	✓
Reusability	-	✓	✓	Θ	-	-	✓	✓	✓

interaction generates and sends SPARQL queries in the background. Yet they are highly data-intensive, which is often impractical for large data sets. Finally, PepeSearch uses conventional forms and mixes schema-based search and browsing; search is limited with a kernel concept and concepts directly related to it, and relevant terms are extracted from the data. Apart from limited expressivity, PepeSearch suffers from poor domain knowledge extracted from data (i.e., compared to a rich ontology), although the interface is naturally tailored by the data. Secondly it does not offer means to cope with large and frequently changing datasets (i.e., one needs to re-extract schema information if data changes).

As far as temporal queries are concerned, notable examples of temporal query languages in the Semantic Web are C-SPARQL [11], SPARQLstream [18], and CQELS [55]. These approaches extend SPARQL with a window operator whose content is a multi-set of variable bindings for the open variables in the query. However, in this paper we are rather interested in visual solutions sitting on top of any of these languages. Although several visual tools exist for SPARQL (cf. [75]), the work is very limited for stream languages. An example is SPARQL/CQELS visual editor designed for Super Stream Collider framework [64]. However, the tool follows the jargon of the underlying language closely and is not appropriate for end users as it will demand considerable knowledge and skills.

Concerning spatial querying, notable formal textual query languages are stSPARQL [13] and geoSPARQL[61]. stSPARQL is an extension of SPARQL 1.1 for querying linked geospatial data that changes over time, while geoSPARQL is a recent standard of the Open Geospatial Consortium (OGC) for static geospatial data. Although there are numerous tools for visualizing and interacting with spatial data such as Sextant [60], visual query tools are limited. A visual query tool is being developed in TELIOS project [28], which is an adaptation from an earlier facet-based search tool by introducing a supplementary map component for constraining certain location dependent attributes.

In Figure 5, a comparison of the prominent tools are given. The summary suggests that none of the tools alone can address our industrial requirements. The majority of tools presented are either formal or have a strong focus on browsing, which leads them to be highly explorative and instance oriented. Browsing being very adequate for open Web, in our context interacting with the ontology, instead of directly with the data, is more suitable for domain experts and compu-

tationally feasible due to the large data size and the nature of the tasks. OptiqueVQS is, however, a visual query system working primarily at a conceptual level and it is not our concern to reflect the underlying formality (i.e., query language and ontology) per se. We are also not interested in providing full expressivity, as we aim to reach a usability-expressiveness balance. The design of OptiqueVQS is based on clear requirements, solid design choices with a rationale, and quality attributes. Finally, there is a lack of rigorous theoretical underpinning in the context of RDF and OWL 2. Existing approaches mostly either focus on RDF, thus essentially disregarding the role of OWL 2 ontologies, or do not reveal how underlying semantics are projected to drive exploration and query formulation.

9. Conclusion and Future Work

In this article, we have proposed an interactive end-user visual query formulation tool OptiqueVQS that is based on a pragmatically motivated requirements and relies on a theoretical framework of navigation graphs. OptiqueVQS is targeted for addressing complex and specific information needs without demanding any specialised IT background and aims at providing a good balance between usability and expressivity. We evaluated our solution with different users groups, including two industrial use cases, with limited IT knowledge and skills with encouraging results.

The future work involves exploring possibilities for realising more complex query types hidden at different layers. However, the aim is not to reach full expressivity, but to implement simpler forms of complex query operations. For example, disjunction and negation at data property level is expected to be easier compared to disjunctions at object property level. Finally, further user studies are planned to validate the core design choices behind OptiqueVQS.

References

[1] OWL 2 Web Ontology Language: Direct Semantics. http://www.w3.org/TR/owl2-direct-semantics/, 2012. W3C Recommendation.

[2] L. Abele, S. Grimm, S. Zillner, and M. Kleinsteuber. An Ontology-Based Approach for Decentralized Monitoring and Diagnostics. In *Proceedings of the 12th IEEE International Conference on Industrial Informatics (INDIN 2014)*, pages 706–712. IEEE, 2014.

[3] L. Abele, C. Legat, S. Grimm, and A. W. Muller. Ontology-Based Validation of Plant Models. In *Proceedings of the 11th IEEE International Conference on Industrial Informatics (INDIN 2013)*, pages 236–241. IEEE, 2013.

[4] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.

[5] O. Ambrus, K. Möller, and S. Handschuh. Konduit VQB: A Visual Query Builder for SPARQL on the Social Semantic Desktop. In *Proceedings of the Workshop on Visual Interfaces to the Social and Semantic Web (VISSW 2010)*, volume 565 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.

[6] J. Arancón, L. Polo, D. Berrueta, F. Lesaffre, N. Abajo, and A. M. Campos. Ontology-Based Knowledge Management In The Steel Industry. In J. Cardoso, M. Hepp, and M. D. Lytras, editors, *The Semantic Web: Real-World Applications from Industry*, pages 243–272. Springer, 2007.

[7] M. Arenas, B. Cuenca Grau, E. Kharlamov, Š. Marciuška, and D. Zheleznyakov. Faceted search over RDF-based knowledge graphs. *Web Semantics: Science, Services and Agents on the World Wide Web*, 37-38:55–74, 2016.

[8] M. Arenas, B. Cuenca Grau, E. Kharlamov, Š. Marciuška, and D. Zheleznyakov. Faceted Search over Ontology-Enhanced RDF Data. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM 2014)*, pages 939–948. ACM, 2014.

[9] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[10] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[11] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, and M. Grossniklaus. C-SPARQL: SPARQL for Continuous Querying. In *Proceedings of the 18th International World Wide Web Conference (WWW 2009)*, pages 1061–1062. ACM, 2009.

[12] G. Barzdins, E. Liepins, M. Veilande, and M. Zviedris. Ontology Enabled Graphical Database Query Tool for End-Users. In *Proceedings of the 8th International Baltic Conference on Databases and Information Systems (DB&IS 2008)*, pages 105–116. IOS Press, 2009.

[13] K. Bereta, P. Smeros, and M. Koubarakis. Representation and Querying of Valid Time of Triples in Linked Geospatial Data. In *Proceedings of the 10th International Conference on the Semantic Web: Semantics and Big Data (ESWC 2013)*, volume 7882 of *LNCS*, pages 259–274. Springer, 2013.

[14] N. Bevan and M. MacLeod. Usability Measurement in Context. *Behaviour & Information Technology*, 13(1-2):132–145, 1994.

[15] C. Bobed, G. Esteban, and E. Mena. Enabling Keyword Search on Linked Data Repositories: An Ontology-based Approach. *International Journal of Knowledge-based and Intelligent Engineering Systems*, 17(1):67–77, 2013.

[16] J. M. Brunetti, R. García, and S. Auer. From Overview to Facets and Pivoting for Interactive Exploration of Semantic Web Data. *International Journal on Semantic Web and Information Systems*, 9(1):1–20, 2013.

[17] P. Brusilovsky, A. Kobsa, and W. Nejdl, editors. *The Adaptive Web: Methods and Strategies of Web Personalization*. LNCS. Springer, 2007.

[18] J.-P. Calbimonte, O. Corcho, and A. J. G. Gray. Enabling Ontology-based Access to Streaming Data Sources. In *Pro-*

ceedings of the 9th International Semantic Web Conference (ISWC 2009), volume 6496 of *LNCS*, pages 96–111. Springer, 2010.

[19] D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao. Ontop: Answering SPARQL Queries over Relational Databases. *Semantic Web*, (in press), 2016.

[20] S. Campinas, T. E. Perry, D. Ceccarelli, R. Delbru, and G. Tummarello. Introducing RDF Graph Summary with Application to Assisted SPARQL Formulation. In *Proceedings of the 23rd International Workshop on Database and Expert Systems Applications (DEXA 2012)*, pages 261–266. IEEE Computer Society, 2012.

[21] T. Catarci. What Happened When Database Researchers Met Usability. *Information Systems*, 25(3):177–212, 2000.

[22] T. Catarci, M. F. Costabile, S. Levialdi, and C. Batini. Visual Query Systems for Databases: A Survey. *Journal of Visual Languages and Computing*, 8(2):215–260, 1997.

[23] C. Civili, M. Console, G. De Giacomo, D. Lembo, M. Lenzerini, L. Lepore, R. Mancini, A. Poggi, R. Rosati, M. Ruzzi, V. Santarelli, and D. F. Savo. MASTRO STUDIO: managing ontology-based data access applications. *PVLDB*, 6(12):1314–1317, 2013.

[24] B. Cuenca Grau, M. Giese, I. Horrocks, T. Hubauer, E. Jimenez-Ruiz, E. Kharlamov, M. Schmidt, A. Soylu, and D. Zheleznyakov. Towards Query Formulation and Query-Driven Ontology Extensions in OBDA Systems. In *Proceedings of the 10th International Workshop on OWL: Experiences and Directions (OWLED 2013)*, CEUR Workshop Proceedings. CEUR-WS.org, 2013.

[25] B. Cuenca Grau, I. Horrocks, B. Motik, B. Parsia, P. F. Patel-Schneider, and U. Sattler. OWL 2: The Next Step for OWL. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):309–322, 2008.

[26] D. Damljanovic, M. Agatonovic, H. Cunningham, and K. Bontcheva. Improving Habitability of Natural Language Interfaces for Querying Ontologies with Feedback and Clarification Dialogues. *Web Semantics: Science, Services and Agents on the World Wide Web*, 19:1–21, 2013.

[27] A. K. Dey. Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[28] U. Di Giammatteo, M. Sagona, and S. Perelli. The TELEIOS software architecture. `http://www.earthobservatory.eu/deliverables/FP7-257662-TELEIOS-D1.2.2.pdf`.

[29] R. G. Epstein. The TableTalk Query Language. *Journal of Visual Languages and Computing*, 2:115–141, 1991.

[30] M. Giese, D. Calvanese, P. Haase, I. Horrocks, Y. Ioannidis, H. Kllapi, M. Koubarakis, M. Lenzerini, R. Möller, M. Rodriguez-Muro, Ö. Özçep, R. Rosati, R. Schlatte, M. Schmidt, A. Soylu, and A. Waaler. Scalable End-User Access to Big Data. In R. Akerkar, editor, *Big Data Computing*, pages 205–244. CRC Press, 2013.

[31] M. Giese, A. Soylu, G. Vega-Gorgojo, A. Waaler, P. Haase, E. Jiménez-Ruiz, D. Lanti, M. Rezk, G. Xiao, Ö. Özçep, and R. Rosati. Optique: Zooming in on Big Data. *IEEE Computer Magazine*, 48(3):60–67, 2015.

[32] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang. Hermit: An OWL 2 reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014.

[33] A. Gliozzo, O. Biran, S. Patwardhan, and K. McKeown. Semantic Technologies in IBM Watson. In *Proceedings of the 4th Workshop on Teaching Natural Language Processing*, pages 85–92. Association for Computational Linguistics, 2013.

[34] I. Grangel-González, L. Halilaj, G. Coskun, S. Auer, D. Collarana, and M. Hoffmeister. Towards a Semantic Administrative Shell for Industry 4.0 Components. In *Proceedings of the IEEE 10th International Conference on Semantic Computing (ICSC 2016)*, pages 230–237. IEEE, 2016.

[35] J. Groppe, S. Groppe, and A. Schleifer. Visual Query System for Analyzing Social Semantic Web. In *Proceedings of the 20th international conference companion on World wide web (WWW 2011)*, pages 217–220. ACM, 2011.

[36] F. Haag, S. Lohmann, S. Bold, and T. Ertl. Visual SPARQL Querying Based on Extended Filter/Flow Graphs. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces (AVI 2014)*, pages 305–312. ACM, 2014.

[37] F. Haag, S. Lohmann, S. Siek, and T. Ertl. QueryVOWL: A Visual Query Notation for Linked Data. In *Proceedings of the Satellite Events of the 12th European Conference on the Semantic Web (ESWC 2015)*, volume 9341 of *LNCS*, pages 387–402. Springer, 2015.

[38] P. Haase, M. Schmidt, and A. Schwarte. The Information Workbench as a Self-Service Platform for Linked Data Applications. In *Proceedings of the Second International Conference on Consuming Linked Data (COLD 2011)*, volume 782 of *CEUR Workshop Proceedings*, pages 119–124. CEUR-WS.org, 2011.

[39] S. Harris and A. Seaborne. SPARQL 1.1 Query Language. http://www.w3.org/TR/sparql11-query/, March 2013.

[40] P. Heim and J. Ziegler. Faceted Visual Exploration of Semantic Data. In *Proceedings of the First IFIP WG 13.7 International Workshop on Human Aspects of Visualization (HCIV 2009)*, volume 6431 of *LNCS*, pages 58–75. Springer, 2011.

[41] A. Hogan, A. Harth, J. Umbrich, S. Kinsella, A. Polleres, and S. Decker. Searching and browsing linked data with swse: The semantic web search engine. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4):365–401, 2011.

[42] F. Hogenboom, V. Milea, F. Fransincar, and U. Kaymak. RDF-GL: A SPARQL-Based Graphical Query Language for RDF. In *Emergent Web Intelligence: Advanced Information Retrieval*, Advanced Information and Knowledge Processing, pages 87–116. Springer, 2010.

[43] M. Horridge, N. Drummond, J. Goodwin, A. L. Rector, R. Stevens, and H. Wang. The Manchester OWL Syntax. In *OWLED*, 2006.

[44] I. Horrocks, O. Kutz, and U. Sattler. The Even More Irresistible $\mathcal{SROIQ}$. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 57–67. AAAI Press, 2006.

[45] A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, and E. Giannopoulou. Ontology Visualization Methods - A Survey. *ACM Computing Surveys*, 39(4):10:1–10:43, 2007.

[46] E. Kaufmann and A. Bernstein. Evaluating the Usability of Natural Language Query Languages and Interfaces to Semantic Web Knowledge Bases. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4):377–393, 2010.

[47] A. Khalili and S. Auer. User Interfaces for Semantic Authoring of Textual Content: A Systematic Literature Review. *Web Semantics: Science, Services and Agents on the World Wide Web*,

22:1–18, 2013.

[48] E. Kharlamov, S. Brandt, E. Jimenez-Ruiz, Y. Kotidis, S. Lamparter, T. Mailis, C. Neuenstadt, O. Ozcep, C. Pinkel, C. Svingos, D. Zheleznyakov, I. Horrocks, Y. Ioannidis, and R. Moller. Ontology-Based Integration of Streaming and Static Relational Data with Optique. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD 2016)*, pages 2109–2112. ACM, 2016.

[49] E. Kharlamov, B. Cuenca Grau, E. Jiménez-Ruiz, S. Lamparter, G. Mehdi, M. Ringsquandl, Y. Nenov, S. Grimm, M. Roshchin, and I. Horrocks. Capturing Industrial Information Models with Ontologies and Constraints: The Siemens Use Case. In *Proceedings of the 15th International Semantic Web Conference (ISWC 2016)*. Springer, 2016.

[50] E. Kharlamov, D. Hovland, E. Jiménez-Ruiz, D. Lanti, H. Lie, C. Pinkel, M. Rezk, M. G. Skjæveland, E. Thorstensen, G. Xiao, D. Zheleznyakov, and I. Horrocks. Ontology Based Access to Exploration Data at Statoil. In *Proceedings of the 14th International Semantic Web Conference (ISWC 2015)*, volume 9367 of *LNCS*, pages 93–112. Springer, 2015.

[51] E. Kharlamov, E. Jiménez-Ruiz, D. Zheleznyakov, D. Bilidas, M. Giese, P. Haase, I. Horrocks, H. Kllapi, M. Koubarakis, Ö. L. Özçep, M. Rodriguez-Muro, R. Rosati, M. Schmidt, R. Schlatte, A. Soylu, and A. Waaler. Optique: Towards OBDA Systems for Industry. In *Proceedings of the 10th International Conference on the Semantic Web (ESWC 2013)*, volume 7955 of *LNCS*, pages 125–140. Springer, 2013.

[52] E. Kharlamov, Y. Kotidis, T. Mailis, C. Neuenstadt, C. Nikolaou, O. Ozcep, S. Christoforos, D. Zheleznyakov, S. Brandt, I. Horrocks, S. Lamparter, Y. Ioannidis, and R. Moller. Towards Analytics Aware Ontology Based Access to Static and Streaming Data. In *Proceedings of the 15th International Semantic Web Conference (ISWC 2016)*. Springer, 2016.

[53] E. Kharlamov, N. Solomakhina, Ö. L. Özçep, D. Zheleznyakov, T. Hubauer, S. Lamparter, M. Roshchin, A. Soylu, and S. Watson. How Semantic Technologies Can Enhance Data Access at Siemens Energy. In *Proceedings of the 13th International Semantic Web Conference (ISWC 2014)*, volume 8796 of *LNCS*, pages 601–619. Springer, 2014.

[54] M. R. Kogalovsky. Ontology-Based Data Access Systems. *Programming and Computer Software*, 38(4):167–182, 2012.

[55] D. Le-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 *Proceedings of the 10th international conference on The semantic web (ISWC 2011)*, volume 7031 of *LNCS*, pages 370–388. Springer, 2011.

[56] H. Lieberman, F. Paternó, M. Klann, and V. Wulf. End-User Development: An Emerging Paradigm. In H. Lieberman, F. Paternó, and V. Wulf, editors, *End-User Development*, volume 9 of *Human-Computer Interaction Series*, pages 1–8. Springer, 2006.

[57] V. Lopez, C. Unger, P. Cimiano, and E. Motta. Evaluating Question Answering over Linked Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 21:3–13, 2013.

[58] J. I. Lopez-Veyna, V. J. Sosa-Sosa, and I. Lopez-Arevalo. KESOSD: Keyword Search over Structured Data. In *Proceedings of the Third International Workshop on Keyword Search on Structured Data (KEYS 2012)*, pages 23–31. ACM, 2012.

[59] G. Marchionini and R. White. Find What You Need, Understand What You Find. *International Journal of Human-Computer Interaction*, 23(3):205–237, 2007.

[60] C. Nikolaou, K. Dogani, K. Bereta, G. Garbis, M. Karpathiotakis, K. Kyzirakos, and M. Koubarakis. Sextant: Visualizing Time-Evolving Linked Geospatial Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 35(1):35–52, 2015.

[61] OGC. Geosparql - a geographic query language for rdf data. `http://www.opengeospatial.org/standards/geosparql`.

[62] O. L. Özcep, R. Möller, and C. Neuenstadt. A Stream-Temporal Query Language for Ontology Based Data Access. In *Proceedings of the 37th Annual German Conference on Artificial Intelligence (KI 2014)*, volume 8736 of *LNCS*, pages 183–194. Springer, 2014.

[63] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *Journal on Data Semantics X*, 10:133–173, 2008.

[64] H. N. M. Quoc, M. Serrano, D. L. Phuoc, and M. Hauswirth. Super Stream Collider: Linked Stream Mashups for Everyone. In *Proceedings of the Semantic Web Challenge 2012 at 11th International Semantic Web Conference (ISWC 2012)*, 2012.

[65] M. Ringsquandl, S. Lamparter, S. Brandt, T. Hubauer, and R. Lepratti. Semantic-Guided Feature Selection for Industrial Automation Systems. In *Proceedings of the 14th International Semantic Web Conference (ISWC 2015)*, volume 9367 of *LNCS*, pages 225–240. Springer, 2015.

[66] M. C. Schraefel, M. Wilson, A. Russell, and D. A. Smith. mSpace: Improving Information Access to Multimedia Domains with Multimodal Exploratory Search. *Communications of the ACM*, 49(4):47–49, 2006.

[67] B. Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *IEEE Computer Magazine*, 16(8):57–69, 1983.

[68] M. G. Skjæveland, M. Giese, D. Hovland, E. H. Lian, and A. Waaler. Engineering Ontology-Based Access to Real-World Data Sources. *Web Semantics: Science, Services and Agents on the World Wide Web*, 33:112–140, 2015.

[69] P. R. Smart, A. Russell, D. Braines, Y. Kalfoglou, J. Bao, and N. Shadbolt. A Visual Approach to Semantic Query Design Using a Web-Based Graphical Query Designer. In *Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2008)*, volume 5268 of *LNCS*, pages 275–291. Springer, 2008.

[70] A. Soylu, P. De Causmaecker, and P. Desmet. Context and Adaptivity in Pervasive Computing Environments: Links with Software Engineering and Ontological Engineering. *Journal of Software*, 4(9):992–1013, 2009.

[71] A. Soylu, P. De Causmaecker, D. Preuveneers, Y. Berbers, and P. Desmet. Formal Modelling, Knowledge Representation and Reasoning for Design and Development of User-centric Pervasive Software: A Meta-review. *International Journal of Metadata, Semantics and Ontologies*, 6(2):96–125, 2011.

[72] A. Soylu and M. Giese. Qualifying Ontology-based Visual Query Formulation. In *Proceedings of the 11th International Conference Flexible Query Answering Systems (FQAS 2015)*, volume 400 of *Advances in Intelligent Systems and Computing*, pages 243–255. Springer, 2015.

[73] A. Soylu, M. Giese, E. Jiménez-Ruiz, E. Kharlamov, D. Zheleznyakov, and I. Horrocks. Towards Exploiting Query History for Adaptive Ontology-based Visual Query Formulation. In *Proceedings of the 8th Metadata and Semantics Re-*

search Conference (MTSR 2014), volume 478 of *CCIS*, pages 107–119. Springer, 2014.

[74] A. Soylu, M. Giese, E. Jimenez-Ruiz, G. Vega-Gorgojo, and I. Horrocks. Experiencing OptiqueVQS: A Multi-paradigm and Ontology-based Visual Query System for End Users. *Universal Access in the Information Society*, 15(1):129–152, 2015.

[75] A. Soylu, M. Giese, E. Kharlamov, E. Jimenez-Ruiz, D. Zheleznyakov, and I. Horrocks. Ontology-based End-user Visual Query Formulation: Why, What, Who, How, and Which? *Universal Access in the Information Society*, (in press), 2016.

[76] A. Soylu, M. Giese, R. Schlatte, E. Jimenez-Ruiz, O. Ozcep, and S. Brandt. Domain Experts Surfing on Stream Sensor Data over Ontologies. In *Proceedings of the 1st International Workshop on Semantic Web Technologies for Mobile and Pervasive Environments*, 2016.

[77] A. Soylu, E. Kharlamov, D. Zheleznyakov, E. Jimenez-Ruiz, M. Giese, and I. Horrocks. Ontology-based Visual Query Formulation: An Industry Experience. In *Proceedings of the 11th International Symposium on Visual Computing (ISVC 2015)*, volume 9474 of *LNCS*, pages 842–854. Springer, 2015.

[78] A. Soylu, F. Modritscher, and P. De Causmaecker. Ubiquitous web navigation through harvesting embedded semantic data: A mobile scenario. *Integrated Computer-Aided Engineering*, 19(1):93–109, 2012.

[79] A. Soylu, F. Moedritscher, F. Wild, P. De Causmaecker, and P. Desmet. Mashups by Orchestration and Widget-based Personal Environments: Key Challenges, Solution Strategies, and an Application. *Program: Electronic Library and Information Systems*, 46(4):383–428, 2012.

[80] A. Soylu, M. Skjæveland, M. Giese, I. Horrocks, E. Jimenez-Ruiz, E. Kharlamov, and D. Zheleznyakov. A Preliminary Approach on Ontology-based Visual Query Formulation for Big Data. In *Proceedings of the 7th International Conference on Metadata and Semantic Research (MTSR 2013)*, volume 390 of *CCIS*, pages 201–212. Springer, 2013.

[81] D.-E. Spanos, P. Stavrou, and N. Mitrou. Bringing Relational Databases into the Semantic Web: A Survey. *Semantic Web*, 3(2):169–209, 2012.

[82] B. Suh and B. B. Bederson. OZONE: A Zoomable Interface for Navigating Ontology Information. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI 2002)*, pages 139–143. ACM, 2002.

[83] A. Sutcliffe. Evaluating the Costs and Benefits of End-user Development. *ACM SIGSOFT Software Engineering Notes*, 30(4):1–4, 2005.

[84] A. H. M. ter Hofstede, H. A. Proper, and T. P. van der Weide. Query Formulation as an Information Retrieval Problem. *The Computer Journal*, 39(4):255–274, 1996.

[85] T. Tran, D. M. Herzig, and G. Ladwig. SemSearchPro - Using Semantics Throughout the Search Process. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4):349–364, 2011.

[86] D. Tunkelang and G. Marchionini. *Faceted Search*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan and Claypool Publishers, 2009.

[87] V. Uren, Y. Lei, V. Lopez, H. Liu, E. Motta, and M. Giordanino. The Usability of Semantic Search Tools: A Review. *The Knowledge Engineering Review*, 22(4):361–377, 2007.

[88] C. J. van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 2 edition, 1979.

[89] G. Vega-Gorgojo, M. Giese, S. Heggestøyl, A. Soylu, and A. Waaler. PepeSearch: Semantic Data for the Masses. *PLoS ONE*, 11(3), 2016.

[90] G. Vega-Gorgojo, L. Slaughter, M. Giese, S. Heggestøyl, A. Soylu, and A. Waaler. Visual Query Interfaces for Semantic Datasets: An Evaluation Study. *Web Semantics: Science, Services and Agents on the World Wide Web*, 39:81–96, 2016.