Universal Access in the Information Society
accepted article / 2015

Experiencing OptiqueVQS: A Multi-paradigm and Ontology-based Visual Query System for End Users

Ahmet Soylu · Martin Giese · Ernesto Jimenez-Ruiz · Guillermo Vega-Gorgojo · Ian Horrocks

Received: date / Accepted: date

Abstract Data access in an enterprise setting is a determining factor for value creation processes, such as sense making, decision making, and intelligence analysis. Particularly, in an enterprise setting, intuitive data access tools that directly engage domain experts with data could substantially increase competitiveness and profitability. In this respect, the use of ontologies as a natural communication medium between end users and computers has emerged as a prominent approach. To this end, this article introduces a novel ontology-based visual query system, named OptiqueVQS, for end users. OptiqueVQS is built on a powerful and scalable data access platform and has a user-centric design supported by a widget-based flexible and extensible architecture allowing multiple coordinated representation and interaction paradigms to be employed. The results of a usability experiment performed with non-expert users suggest that OptiqueVQS provides a decent level of expressivity and high usability, and hence is quite promising.

Keywords Visual query formulation · visual query systems · ontology-based data access · data retrieval

1 Introduction

A tremendous amount of data is being generated every day both on the Web and in public and private organisations; IBM estimates that 40 Zettabytes of data will be created by 2020 (an increase of 300 times from 2005) and 2.5 Quintillion bytes of data are created each day, while most companies in U.S. have at least 100 Terabytes of data stored[1]. By all accounts, any individual or organisation who possesses necessary knowledge, skills, and tools to make value out of data at such scales, bears a considerable advantage. In an organisational setting, ability to access and use data in business processes such as *sense making* and *intelligence analysis* is key for the *value creation* potential (cf. [64]). Today, however, *data access/data retrieval* (cf. [1]) still stands as a major bottleneck for many organisations. One of the key reasons is the sharp distinction between employees who have technical skills and knowledge to extract data from databases (i.e., *database/IT experts*, *skilled/expert users*) and those who lack technical skills but have domain knowledge and know how to interpret and use data (i.e., *domain experts*, *non-skilled/non-expert users*). The result is a workflow where domain experts either have to use pre-defined queries embedded in applications or communicate their *information needs* to database experts, who in turn translate them into formal queries possibly over disparate data sources. The former scenario is quite limiting, since it is not possible to enumerate every possible information need beforehand, while the latter scenario is hampered by the ambiguity in communication. In such a workflow, the turn-around time from the initial information need of a user to receiving an answer can be in the range of weeks, incurring significant costs (cf. [33]). Therefore, for instance in an enterprise setting, engaging domain experts directly with data could substantially increase competitiveness and profitability.

Approaches that eliminate the *man-in-the-middle* and allow end users who have no technical skills to di-

Ahmet Soylu
Gjøvik University College, and University of Oslo
E-mail: ahmet.soylu@hig.no, ahmets@ifi.uio.no

Martin Giese · Guillermo Vega-Gorgojo
Department of Informatics, University of Oslo

Ernesto Jimenez-Ruiz · Ian Horrocks
Department of Computer Science, University of Oxford

[1] `http://www.ibm.com/big-data/us/en/`

rectly engage with data and extract it on their own have been of interest to the researchers for many years now (cf. [17]). As anticipated, for end users, the accessibility of traditional structured query languages such as *SQL* and *XQuery* fall far short, since formal textual languages do require end users to have a set of technical skills and to *recall* domain concepts, and the terminology and syntax of the language being used. *Visual query systems and languages* (VQSs and VQLs) (cf. [17]) emerged to alleviate the end-user data access problem by providing intuitive and natural end-user experiences. A visual system or language follows the *direct-manipulation* idea (cf. [74]), where domain and query language are represented with a set of visual elements. End users *recognise* the relevant fragments of domain and language and formulate queries basically by directly manipulating them. A visual approach increases *user engagement* and enables users to easily *digest/grasp*, *communicate*, and *interact* even with larger amounts of information (cf. [88, 44]), hence are more *accessible* (cf. [26,56]). A good deal of research on visual query formulation exists both for *structured* (e.g., relational data) and *semi-structured data* (e.g., XML), such as *QBE* [93] and *Xing* [28]. Although early approaches (cf. [17,16]) successfully establish the research field, their success, in practical terms, remains within the confines of abstraction levels they operate on; database schemas, object-oriented models, etc. are not meant to capture a domain per se and are not truly natural for end users (cf. [75,71,42]). The use of *ontologies* as a natural communication medium for end users appeared as a prominent approach; however, early attempts on ontology-based visual query formulation (e.g., [18,3]) did not develop much and remained at experimental stages mostly due to the lack of appropriate frameworks that bridge ontologies and relational data sources. It goes without saying that today relational databases accommodate the most of world's enterprise data; nevertheless, the gap is almost closed with the *Semantic Web* (cf. [6,8]) and *ontology-based data access* (OBDA) technologies (cf. [69,54,63]), which enable access to legacy relational data over ontologies.

The aforementioned advances led to the reappraisal of the role that ontologies have to play for visual query formulation. There exist numerous works on ontology-based/semantic search interfaces (e.g., [18,3,40,41]). However, they either focus on *browsing* rather than *querying* and hence are largely instance oriented, or are attached to one predominant *interaction* and/or *representation paradigm*, and hence are very limited in terms of usability and expressivity, or lack adequate architectures, with which a query tool can be extended and enriched as needed. These are mostly due to design approaches that focus on functionality concerns rather than usability. Yet, the challenge is mostly one of usability, which should therefore steer and dominate any functionality consideration. An ongoing EU project, named *Optique – Scalable End-user Access to Big Data*[2], promises to deliver a platform that provides an *end-to-end solution* for scalable end-user data access both in terms of query formulation and query answering through a visual query interface and an OBDA framework. The platform bridges ontologies and relational databases and is supported by other components such as for *query parallelisation*, *optimisation*, *distributed query execution*, and *time and stream management* (cf. [33]).

This article, having Optique as a motivating scenario, is concerned with ontology-based visual query formulation for querying relational databases for end users with no technical skills and knowledge, such as on programming, databases, query languages, and with low/no tolerance, intension, or time to use and learn formal languages. The primary contributions presented in this article are a novel and easy-to-use concept and a flexible and extensible widget-based architecture for an ontology-based VQS, based on multiple coordinated representation and interaction paradigms for graph navigation and facet refinement, along with a prototype named *OptiqueVQS* [77,81]. The design of the OptiqueVQS is guided through industrial use cases provided by two large energy companies, namely *Statoil*[3] and *Siemens*[4]. The article discusses the expressivity of the OptiqueVQS, presents the results of the first usability study, and identifies a set of open research challenges.

The rest of the article is structured as follows. Section 2 presents and discusses the motivating scenario and data access interfaces. Section 3 presents the background and related work on VQSs, while Section 4 sets a set of requirements for expressiveness and usability. Section 5 and Section 6 describe the OptiqueVQS approach and its implementation respectively. Section 7 presents the usability experiment, while Section 8 provides a discussion. Finally, Section 9 concludes the article.

2 Research context

The capacity to find, access, and process data forms the *intellectual bandwidth* of an organisation, which is leveraged through appropriate methodologies for value creation (cf. [64]). The value creation potential of an organisation is dependent on how data is stored, accessed and used, in other words, on how an organisation utilises its *data*, *hardware*, *software*, and *human* capitals, not

[2] `http://www.optique-project.eu`
[3] `http://www.statoil.com`
[4] `http://www.siemens.com`

only individually but as an *ecosystem* (e.g., collaboration and interoperability). Data access is one of the principal components of intellectual bandwidth and today stands as a major bottleneck in many organisations; domain experts spend considerable amount of time on data access problems, which could be redeployed so as to lead to even greater value creation (cf. [33, 24]).

A *data access system* requires interfaces that elicit the information needs of users and transform them into formal queries, and a backend system that manages data sources and evaluates queries against them, which are, in many cases, structured and are described through a data model. This section provides an overview of the Optique platform to demonstrate the role of visual query systems and languages in data access and to introduce key technologies for the practical applicability of ontology-based visual query formulation and data access in general. Then, different types of data access interfaces are presented and compared.

2.1 Motivating scenario: Optique

The Optique platform addresses end-user data access problems primarily by bringing a visual query system together with an OBDA framework for relational data sources, which leads to a semantic end-to-end connection, between end users and data sources, allowing maximum data exploitation. The visual query formulation tool is meant to enable end users to rapidly formulate queries using familiar vocabularies and conceptualisations, while the OBDA framework is to ensure seamless integration of data, spread across multiple distributed data sources including *streaming*, *temporal*, and *spatial* ones, and is powered with query optimisation and massive parallelism. Ontologies not only act as a *super structure* over distributed data sources (i.e., *federation*) and as a natural communication medium, but also provide *reasoning* support, which is precious both at query formulation and answering stages. This is because reasoning provides the capability of expressing more with less by relating the whole set of implied information instead of what is explicitly stated and available (cf. [84]).

The overall approach is depicted in Figure 1; briefly, end users interface with the system through a visual query formulation tool, which also supports IT experts through a textual editor. The tool relies on an OBDA framework that allows access to underlying data sources over ontologies. Once a visual query is translated into a textual form (e.g., SPARQL), it is passed through two *rewrite* phases to transform it into a *complete*, *correct*, and *highly optimised* query over data sources (cf. [69, 68]). The first phase rewrites the query by taking ontological constraints into account, while the second one translates the query into the language of underlying data sources (e.g., SQL) through *mappings* (cf. [82, 69]) that relate the concepts and relations of the ontology to data sources. The mapping approach separates *transactional* and *domain perspectives* (cf. [61]), in other words, while exploiting ontologies for data access and reasoning, one can continue to use legacy relational data sources in their original form, without migrating or transforming any data, and enjoy the benefits of well established query optimisation and evaluation support available for traditional database management systems. In order to aid ontology and mapping development, a *bootstrapping* component (cf. [73]) automatically harvests existing schemas and ontologies and generates an initial ontology and mappings; this is followed by a manual fine-tuning and enhancement process.

The OBDA framework and other components that form the *Optique platform* are beyond the scope of this article; interested readers are referred to Giese et al. [33] and Kharlamov et al. [50].

2.2 Data access interfaces

One could distribute well-known *data access interfaces* for querying structured data into five categories, namely *formal textual languages*, *keyword search*, *natural language interfaces*, *visual query languages*, and *visual query systems*. A VQL is visual programming language based on well-defined formal semantics with a visual notation and syntax, while a VQS is a system of interactions that generate queries (cf. [27]).

A qualitative summary of these approaches are given in Table 1 with respect to *expressiveness* and *usability*. Usability and expressiveness are the two sides of a coin, when it comes to the evaluation of a data access interface. Usability is mainly defined as the extent to which a tool is competent of meeting its identified aim with *effectiveness*, *efficiency*, and *user satisfaction* (i.e., user attitude such as trust, engagement, and acceptance) (cf. [7]) – among the other aspects of usability such as *accessibility* and *learnability*. In this context, effectiveness refers to the ability of a query interface to translate user inputs into complete and accurate queries, while efficiency refers to time and effort required for users to complete a task (cf. [7]). Learnability is the capability of a tool to enable users to learn how to use it, while accessibility[5] is the degree to which a tool is available to as many people as possible. Accessibility and learnability are integral parts of usability in this context,

[5] Though accessibility often focuses on people with disabilities or special needs, in the present context with a broader interpretation, it relates to knowledge and skill barriers.

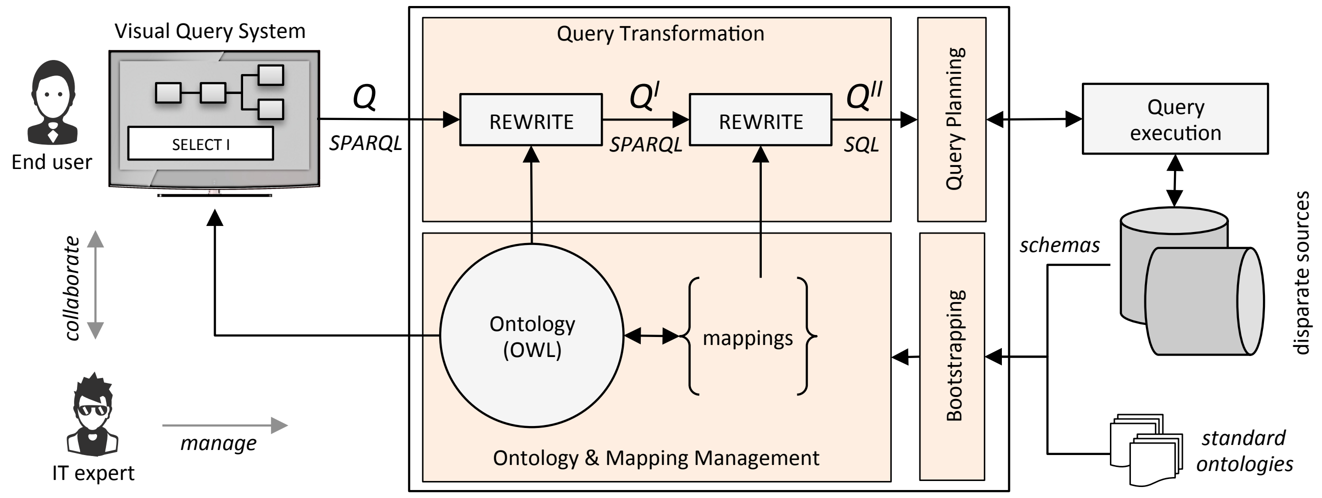


Fig. 1 Visual query formulation and ontology-based data access.

since end-user query formulation prominently demands more skills and knowledge, introducing comprehensibility and availability barriers. Finally, expressiveness is the breadth of a data access interface in characterising a domain and information need (cf. [27]).

Formal textual languages typically allow users to express their information needs completely and accurately with high expressivity in a programmatic way; however, their accessibility and learnability are quite low, since they demand expert knowledge and skills – though some textual query editors assist users by suggesting terms and concepts (e.g., [15]). Keyword search (e.g., [9]) interprets a query as a bag of words, hence is the worst when it comes to completeness, accuracy and expressiveness. There exists studies for increasing completeness and accuracy of keyword search, so that it can be used over structured data (e.g., [59]); however, the inherent limits of keyword search are still valid. Nevertheless, keyword search stands as a good option in terms of accessibility, learnability and efficiency, since it does not demand any expert knowledge and skills. Natural query interfaces (e.g., [48,91]) go beyond the keyword search; this is due to the fact that they interpret a query as whole and take linguistic considerations into account. However, they do not achieve much in terms of accuracy and completeness due to the ambiguity of natural language; some approaches try to alleviate this issue, such as by acquiring feedback and clarification through user dialogues (e.g., [25,58]), yet this increases the user cognitive load. Finally, visual query languages and systems (cf. [17]), which pursue the direct manipulation idea, come particularly with high effectiveness and expressiveness; however, visual query languages (e.g., [29, 55,76,31,43]) usually perform badly in terms of accessibility and learnability, because users still have to respect syntactic and semantic constraints.

Aforementioned approaches have varying degrees of achievement in each dimension; however, their successes depend on the context of use. On the one hand, for instance, although keyword search performs worst in terms of effectiveness, the high tolerance of the tasks in the Web to low accuracy and completeness makes accessibility and learnability aspects predominant for the usability and success of keyword search. On the other hand, unlike *information retrieval* in the Web, data retrieval in traditional database systems heavily relies on complex structures and semantics with no tolerance to irrelevant and missing results and vaguely described information needs (cf. [1]). Therefore, it is concluded that VQSs are a viable option for end-user querying structured data, since they can effectively hide the complexity of a domain, hence demanding low expert knowledge and skills, and provide high expressivity, completeness, and accuracy.

3 Visual Query Systems

Visual query formulation based on ontologies is indeed a multi-front endeavour situated on a set of interrelated research challenges, such as *ontology-driven information systems* (more specifically *ontology-driven user interfaces*) (cf. [71,4]), *visual computing* (more specifically *visual programming*) (cf. [13,12]), and *end-user development/programming*[6] (cf. [57]).

[6] End-user development is defined as a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, to create, modify, or extend software artefacts [57]; and forms the definition of end user in the context of this article.

Table 1 A qualitative comparison of data access interfaces for querying structured data.

Approach/ Criteria	Effectiveness (Completeness & Accuracy)	Efficiency (Time & Effort)	Accessibility & Learnability	Expressiveness
Formal languages	⋆ ⋆ ⋆ ⋆ ⋆	⋆	⋆	⋆ ⋆ ⋆ ⋆ ⋆
Keyword search	⋆	⋆ ⋆ ⋆ ⋆ ⋆	⋆ ⋆ ⋆ ⋆ ⋆	⋆
Natural language interfaces	⋆ ⋆	⋆ ⋆ ⋆	⋆ ⋆ ⋆ ⋆	⋆ ⋆ ⋆ ⋆ ⋆
Visual query languages	⋆ ⋆ ⋆ ⋆ ⋆	⋆ ⋆	⋆ ⋆	⋆ ⋆ ⋆ ⋆ ⋆
Visual query systems	⋆ ⋆ ⋆ ⋆ ⋆	⋆ ⋆ ⋆ ⋆	⋆ ⋆ ⋆ ⋆	⋆ ⋆ ⋆ ⋆

Therefore one needs to address a variety of issues ranging from the formal aspects of the underlying representation paradigm and its visual representation to overall visual coherence and affordances provided for the interaction, as discussed in the following.

3.1 Background

A VQS could use a VQL, could be built on ad-hoc representations, or could directly generate queries in target formal textual form; however, in any case, it is the interactions that matter rather than the visual formalism. Nevertheless, there are two types of activities that a visual query formulation tool needs to support for end users, namely *exploration* (aka *understanding the reality of interest*) and *construction* (cf. [17]). Exploration is the process of understanding the conceptual space describing the domain of interest, while the construction is the process of formalising the information need through the elements of the underlying conceptualisation.

The success of a visual query formulation tool is grounded on how appropriately the following three notions are selected and intertwined (see Figure 2): *visual representation paradigms* (e.g., diagrams and forms), *interaction paradigms* (e.g., navigation and range selection), and *visual attributes and metaphors* (cf. [17]). These notions refer to visual constructs that correspond to domain elements, the way users manipulate the visual objects and hence interact with the system, general visual characteristics of the tool (e.g., size and colour), and analogies with the real life objects, situations, processes, etc. respectively. The ultimate goal is the maximum exploitation of the *human visual channel* and *cognitive capacity*, while demanding less *skills*, such as *communication* and *motor*, and *knowledge*, such as *semantic* and *syntactic*, from users (cf. [74, 90]). These basically include non-volatile semantic knowledge acquired through general explanation, analogy and example, and common skills for manipulating physical objects.

A VQL usually has one-to-one correspondence with the underlying formality, while a VQS is often intentionally kept less expressive. This is firstly due to distinct goals of a VQL and VQS; the former is mainly meant to be a visual replica of the underlying textual language, while the latter is largely meant to generate fragments of it without necessarily reflecting any formal aspects to users. Secondly, there is trade-off between expressiveness and usability, which commonly necessitates sacrificing some complex query and domain constructs, which are hard to deal with in visual form, in the favour of usability. In this regard, particularly for a VQS, a key concern is to reach a usability and expressiveness balance.

3.2 Related work

Early examples of VQSs, as mentioned earlier, built on low level models (cf. [17]), while recent ontology-based approaches mostly target web data (i.e., *linked data*) (cf. [8]). End users generally have a better affinity with data access approaches available for the Web than the ones available for traditional database systems. Their apparent success makes it a sensible direction to adapt them for relational data sources. In this regard, *Faceted search* (cf. [89]) and *Query by Navigation* (QbN) (cf. [87]) are prominent in terms of their suitability for ontologies and their inherent compatibility to each other.

3.2.1 Faceted search and QbN

Faceted search, being an advanced *form-based* approach, is based on a series of orthogonal dimensions, which can be applied in combination to filter the information space; each dimension, called *facet*, corresponds to a taxonomy. Typically, each facet is derived from the properties of the corresponding concept. In its most common form, facet options are accompanied with the number of accessible instances, called *numeric volume indicators* (NVI), for each possible selection to prevent users from reaching empty result sets. A track of user selections, called *bread crumbs*, are also kept for user awareness and manipulation. QbN exploits the graph-based organisation of information to allow users to construct queries by traversing the conceptual space through relationships between concepts; each pass from one concept to another indeed corresponds to a join operation. End users

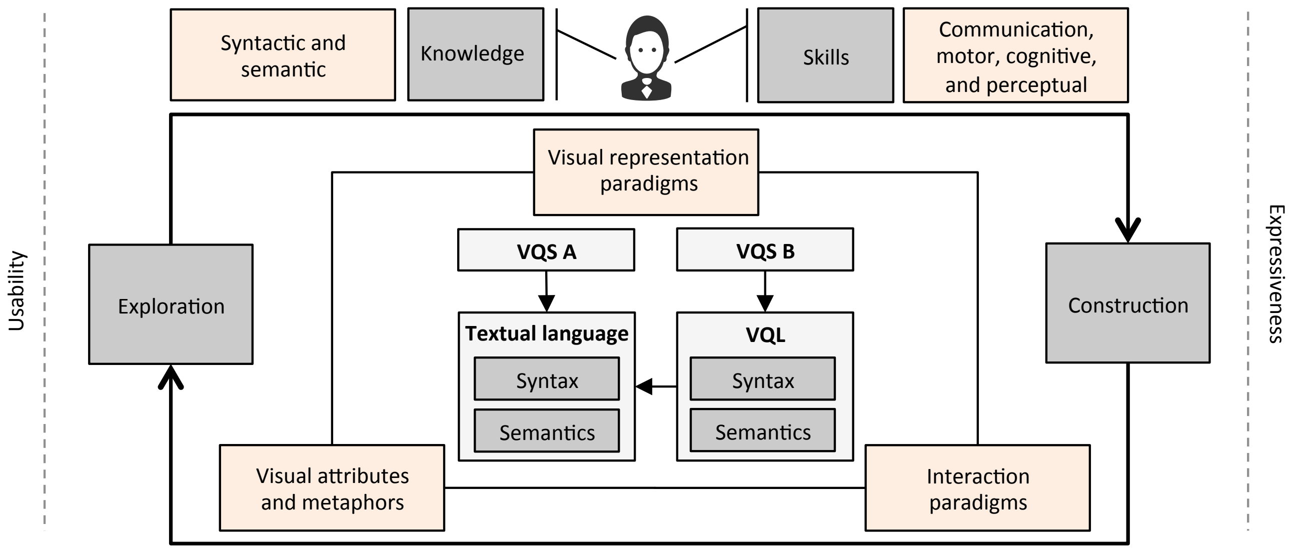


Fig. 2 A conceptual view on end users and visual query formulation.

are quite familiar with both types of search approaches; faceted search is widely used in commercial websites such as eBay and Amazon for listing and filtering products, while the navigation is the backbone of web browsing.

Examples of QbN are *Tabulator* [5], a linked data browser; *SWC* [79], a data navigation tool for the Semantic Web; *SEWASIE* project [18], an ontology-based query formulation tool; *Visor* [66], a data exploration tool for the Semantic Web; and *ViziQuer* [94], a tool for exploring and querying SPARQL end-points. QbN is either a predominant or sole paradigm in these tools. Well-known examples of faceted search are *Flamenco* [92], a faceted search interface for images – not ontology-based; *mSpace* [72], a faceted search framework for the Semantic Web; and *Exhibit* [46], a lightweight framework for publishing structured data. Examples of faceted search are quite typical and well accepted by end users; however, faceted search interfaces commonly allow filtering on a single concept and breaks down as soon as a "join", which is a harder operation for end users to comprehend, between information about several objects is required.

SEWASIE, ViziQuer, and Visor provide a diagram-based pane, where selected concepts appear as nodes, and they support selection and join through form-based dialogs, which are not primarily visible. In SEWASIE and ViziQuer navigation starts with a single concept (i.e., *kernel*), while in Visor a user can start with multiple concepts. In order to support end users, ViziQuer allows users to draw links between any two nodes, so that the system can recommend connections. A more advanced form of similar support is provided by Visor, where users can select multiple concepts, initially unconnected, and then the system finds and suggests links connecting all the nodes. Both in ViziQuer and Visor, concepts are not necessarily required to be directly connected, intermediary nodes can also be found and a path can be suggested – mostly the shortest path. Although this type of support is quite important, particularly for large ontologies and connecting concepts that are multiple hops away from each other, a plain shortest path approach, in many cases, is not expected to lead to sensible connections.

3.2.2 Hybrid approaches

Generally, the examples of QbN provide weak or no support for *select and projection operations*; similarly the examples on the faceted search provide weak or no support for *join operations*. The combination of QbN and faceted search is a promising direction, since there is a fair share of primary query construction operations, i.e., select and project for faceted search and join for QbN. The hybrid of QbN and faceted search is available in two forms in the literature. The first one is built on *menu-based QbN*; the prominent examples are *Parallax* [45], a set-based browsing application for the Web; *Humboldt* [53], a user interface for browsing RDF data; *VisiNav* [40], a system for search and navigation of web data; and *tFacet* [10], a tool for hierarchical visual exploration of semantic data. The second hybrid form is built on *diagram-based QbN*; the prominent examples are *OZONE* [86], an interface for navigating ontology information, and *gFacet* [41], a tool for visual exploration of semantic data.

In menu-based QbN with faceted search, the process is often initiated with a keyword search and then a set of matching concepts and instances are displayed. After the selection of a concept, a set of facets are displayed along with a set of domain concepts that are related to the current concept. In Humboldt, rather than deriving a facet for each relationship, only a single facet is derived per associated concept. Such an approach decreases the number of facets to be displayed; however, it also introduces ambiguity problems. In tFacet, facets are displayed in a tree form; the first level of the tree represents the facets derived from the direct relationships and attributes of the base concept, while the children of a tree item represent the facets of another concept associated through the relationship that its parent tree item represents. However, for large ontologies the depth of the tree is an issue, since each tree item would probably have a high number of children, which in turn hinders the usability. An important problem with menu-based hybrid approaches is poor support for providing an *overview* of the overall task (i.e., a global view of connected concepts, constraints imposed, and attributes selected for the output).

In a diagram-based QbN with faceted search, concepts are represented as nodes in a graph and a user navigates by expanding and retracting nodes. Moving from one concept to another changes the focus (i.e., *pivoting*) and the user can impose constraints on the active concept (e.g., *pivot*) by selecting options within each facet. Both in OZONE and gFacet, the search process starts with the selection of a kernel concept from a hierarchical list of concepts and through a keyword search respectively; the selected concept appears in a graph-based pane as a rectangle and facets constitute the inner content. Users can expand relationships into new nodes by selecting the desired relationship from a menu-item attached to each node. In diagram-based hybrid approaches, a better overview is provided, since diagram-based approaches are naturally good at providing a global view. However, this time the problem becomes poor support for *view* (i.e., focus), that is ability to channel user to a specific part of an active task.

Finally, a common problem for both type approaches is their inability to aggregate information from different concepts. Usually the attributes of kernel concept form the output; although in many cases users can change the kernel concept, they cannot select output attributes from different concepts.

3.2.3 Discussion

A general problem with the presented tools is the utilisation of only one predominant representation or interaction paradigm. Usually, one representation or integration paradigm is not sufficient to accommodate different functionality and usability concerns; Catarci et. al [17] suggest that interfaces combining different types of visual representation and interaction paradigms are more promising. Another problem is the lack of *flexible* and *extensible* architectures, which ensure the sustainability of the interface with respect to changing requirements.

The majority of tools presented have a strong focus on browsing, which leads them to be highly explorative and instance oriented. This being very adequate for open Web, in the present context represents a set of issues.

Firstly, user navigation in a conceptual space is mostly for data browsing purposes; a final query, which encompasses the visited concepts, is not generated. Note that, albeit related, browsing and querying are different notions.

Secondly, there is no clear distinction between explorative and constructive user actions and there is a lack of support for view (i.e., the active phase of a query task) and overview (i.e., the general snapshot of a query task).

Thirdly, a frequent interaction with the data is required (i.e., database-intensive). Indeed, supporting end users with cues (cf. [72]) and statistics concerning the underlying data is an effective method; however, scalability issues have to be addressed for large instance sets.

Finally, most of these tools extract domain knowledge from the instance sets rather than an ontology. This leaves end users with poor domain knowledge, since the amount of domain knowledge inferable from instances is very limited.

4 Design requirements

A VQS should primarily match the level of users and should drive the capabilities of the output medium and human visual system at an optimum level, while bridging the gap between the domain representation and user mental model. One first has to decide what type of domain knowledge and information needs are to be addressed. This, in turn, concerns the selection of ontology constructs and semantics that should be communicated at the interface level and the type of queries that end users should be able to formulate. A sustainable design and evolution plan is required to establish and maintain a balance between usability and expressivity during the life cycle of the tool.

Although a VQS could potentially address a broad spectrum of users, the primary concern in this article is end users who have no technical skills and knowledge and need to extract data from databases without ever needing to know a formal query language. These could be *casual users* who occasionally use computers for entertainment and basic tasks in their daily life/work (e.g., surfing on the Web and checking e-mails), or could be *domain experts* who have an in-depth knowledge and understanding of the semantics of their expertise domain and use computers for their work. *SPARQL* [39] is assumed as the query language, and *OWL 2* [36] as the ontology language in the present case. In what follows, the level of expressiveness (i.e., *functional requirements*) and core *quality attributes* (i.e., *non-functional requirements*) required in order to meet the goal of this research are discussed.

4.1 Expressiveness

Expressiveness is addressed from the point of view of exploration and construction.

4.1.1 Exploration

Two forms of exploration are distinguished, namely *passive* and *active exploration*. The former refers to the implicit flow of domain knowledge to end users during query formulation and is an integral part of a VQS. The latter refers to the explicit acquisition of domain knowledge by end users purely for exploration purposes, such as traversing a conceptual space through an ontology visualisation component (cf. [47]).

An ontology is as a *formal*, *explicit* specification of a *shared* conceptualisation, which is an abstract representation of a phenomenon in the world based on identification of relevant concepts, attributes, relationships, and constraints (cf. [85]). Considering a typical ontology, the following ontology constructs should be delivered visually to aid users: *classes* (i.e., concepts), *class hierarchy*, *object type properties* (i.e., relationships), *data type properties* (i.e., attributes), *multiple inheritance*, *enumerated classes*, *inverse relationships*, *multiple ranges*, *disjointness*, and *subproperties*. The remaining constructs, such as *role chains* and *transitivity*, are not deemed to be necessarily visualised, since they are rather valuable at the query answering stage in terms of *classification*, *inference*, and *consistency checking*. Nonetheless, multiple inheritance, disjointness, subproperties, inverse properties, and multiple ranges are comparatively harder to communicate, while others are mostly well established as far as visualisation methods and tools are concerned (cf. [47]).

Regarding reasoning, the propagation of property restrictions also has to be considered and is twofold: *top-down propagation of property restrictions* and *bottom-up propagation of property restrictions* (cf. [35]). These are based on the facts that an ontology class inherits property restrictions of its parent classes and includes interpretations of all its subclasses. Therefore for a given ontology class, the properties of its subclasses and superclasses can also be suggested.

An ontology alone is not sufficient to successfully communicate domain knowledge. It often needs to be enriched with visualisation related information, such as labels and descriptions for ontology elements. The common practice in the literature is to use concept, relationship, and attribute identifiers as they appear in the ontology; however, these identifiers are not meant for end-user consumption and could deliver a sense of complexity.

4.1.2 Construction

Queries could be classified with respect to their *topological* form into three categories, namely *linear queries*, *queries with branching*, and *cyclic* queries. Linear queries refer to queries only with a sequential path expression (i.e., serial joins of concepts), while queries with branching include branching path expressions combined through "AND" and "OR" connectives (cf. [34,19]). Given a query graph pattern, there exists a cycle, if there is at least one (*undirected*) path in the graph whose first node corresponds to the last. Queries including disjunction and cycles are more problematic compared to the linear and tree-shaped conjunctive queries (cf. [17]).

From a non-topological perspective, queries with *quantification*, *negation*, and *aggregation* are notable (cf. [63,84]). Two fundamental forms of quantification are *existential quantification* (i.e., "there exists") and *universal quantification* (e.g., "for all"). Negation is an operation that may be applied on a proposition, truth value, etc. and, in the simplest terms, is used to reverse a condition, while aggregation functions such as *count*, *sum* and *max* are used to group values of multiple attributes to form a single value. While the use of existential quantification remains implicit, queries that include universal quantification, negation, and aggregation are quite esoteric for end users; this even applies to expert users, particularly for universal quantifiers [49].

Based on the aforementioned categorisations, queries could be prioritised in three levels with respect to the *user perceived complexity* and need:

- *First level*: linear and tree-shaped conjunctive queries (i.e., queries with branching and without cycles);
- *Second level*: disjunctions, cycles and aggregation;
- *Third level*: universal quantifiers and negation.

The first level is considered as the *ground challenge* and is the primary goal in the context of this article.

This is justified with a set of example end-user queries delivered by one of the use case partners, where 80 percent of queries fall into the first level.

4.2 Quality attributes and features

As suggested in the discussion of related work, *usability* and *modularity* are considered as primary and interrelated quality attributes. A set of features that support these attributes are listed in the following.

4.2.1 User-friendly design

Generally, the design should be *intelligible*, *intuitive*, *succinct*, and *stimulative*. More specifically, a visual query formulation tool has to be *instant*, i.e., the result of user actions should immediately be reflected; *gradual*, i.e., particularly for large ontologies and query tasks, users should be able to gradually explore and construct; *reversible*, i.e., during query construction a user typically does several explorative actions (e.g., add/remove constraint), which requires every query state to be recoverable; and *iterative*, i.e., often a query is only complete after several iterations, therefore users should be allowed to modify and use the existing queries.

4.2.2 Multi-paradigm

Various types of representation (form-based, diagram-based, icon-based, etc.) and interaction paradigms (navigation, range selection, etc.) exist in the literature (cf. [17]). One should be aware that not every representation and interaction style goes well with every type of domain and query construct and affordance (cf. [47]). Therefore, a visual query formulation tool should harmonise multiple representation and interaction styles, each best suited for particular type of constructs and affordances. A multi-paradigm approach is promising to address a broad range of tasks and user types.

4.2.3 View/Overview

End users should feel and be in full control of the tool and have continuous awareness of the active state (i.e., *perceived control* and *user situation awareness*); this is expected to have a major effect on the attitude of end users (cf. [83]). In a visual query formulation scenario, the balance between view and overview is the primary contributor for such a control and awareness. A persistent query overview and view empower a user to have an overall understanding of the task at hand and provide ability to switch to and focus on different parts of it at any moment.

4.2.4 Exploration/Construction

Exploration and construction have adverse yet complementary roles. In exploration, a user aims to navigate a conceptual space as broadly as possible, while in construction the goal of user is only to traverse the part of the conceptual space that corresponds to his/her information need with as few deviations and backtracks as possible. Therefore explorative and constructive facilities should be intertwined and supported with due diligence, allowing smooth and frequent transitions between each phase.

4.2.5 Modular architecture

A modular architecture ensures flexibility and extensibility, so that new components could easily be introduced in order to adapt to changing requirements. This could include alternative/complementary components for query formulation, exploration, visualisation, etc. It also underpins the sustainability and a multi-paradigm design, as each component could employ a different representation and interaction paradigm.

5 OptiqueVQS approach

The requirements presented previously are foundational for the OptiqueVQS approach. This section describes the design and design rationale behind the OptiqueVQS interface, after briefly introducing the key technical concepts, which are to be described in detail in Section 6.

OptiqueVQS is designed as a *widget-based user-interface mashup* (i.e., UI mashups); an UI mashup aggregates a set of applications in a common graphical space, in the form of *widgets*, and *orchestrates* them for achieving common goals (cf. [80]). In the present context, widgets are the building blocks of UI mashups and refer to *portable*, *self-contained*, *full-fledged*, and mostly *client side* applications with less functionality and complexity. In a query formulation scenario, a set of widgets could be employed together. For instance, a widget for QbN and a widget for faceted search could handle query construction synchronously, and another widget could present query results.

5.1 Interface

Initially there are three widgets in the design as depicted in Figure 3. The first widget (*W1* – see the bottom-left part of Figure 3) is a menu-based QbN widget and allows users to navigate concepts through pursuing relationships between them, hence joining relations in a database. The second widget (*W2* – see the bottom-right part of Figure 3) is a form-based widget, which presents the attributes of a selected concept for selection and projection operations. The third widget (*W3* – see the top part of Figure 3) is a diagram-based widget and provides an overview of the constructed query and affordances for manipulation. These three widgets are orchestrated by the system, through harvesting event notifications generated by each widget as a user interacts, so that they could jointly extract and represent the information need of the user.

In a typical query construction scenario, a user first selects a kernel concept, i.e., the starting concept, from W1, which initially lists all domain concepts accompanied with icons, descriptions, and the potential/approximate number of results. The selected concept appears on the graph (i.e., W3) as a *variable-node* and becomes the focus/pivot/active node (i.e., the node coloured in orange or highlighted). W2 displays its attributes in the form of text fields, range sliders, etc. The user can select attributes to be included in the result list (i.e., using the "eye" button) and/or impose constraints on them through form elements in W2. Currently, the attributes selected for output appear on the corresponding variable-node in black with a letter "o", while constrained attributes appear in blue with letter "c". 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. Note that once there is a pivot node, W1 does not purely lists concepts anymore but a set of (sub)paths. Each item/path in W1 represents a combination of a possible relationship with its range concept pertaining to the pivot (i.e., indeed a path of length one). The user can select any available item from the list; this results in new path with a new variable-node of type specified by the selected item, a join between the pivot and the new variable-node over the specified relationship, and a move in focus to the new variable-node. 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; this allows queries to be always read from left to right. In W3, it is preferable to employ node duplication approach for cyclic queries for the sake of having tree-shaped query representations, hence avoiding a graph representation. An example query is depicted in Figure 3 for the Statoil use case. The query asks for all fields that has facility, with a water depth of "305,25" meters, and are operated by a company, which also operates a field named "COD". In the output, one would like to see the names of the fields and the short names of the companies (first "Field" variable-node is the kernel, while the second one is the pivot in the snapshot).

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 can also switch to 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 4. 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. Currently, the textual SPARQL query is not manually editable and is for didactical purposes only. More advanced 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.

5.2 Expressiveness

The expressiveness of a visual query language or system concerns the underlying formal ontology and query language (i.e., formal limits) and the fragments of them represented visually (i.e., what and how). In the following, the visual expressiveness of the OptiqueVQS is discussed from exploration (i.e., what it is able to communicate) and construction perspectives (i.e., what it allows users to communicate) and the details of underlying formalisms are given in Section 6.

A VQS is not expected to be fully expressive; this is due to the fact that advanced query constructs, even in visual form, could be hard to comprehend and use for end users, while for IT experts textual mode would probably be more efficient and comfortable. In this respect, only domain and query constructs, which are frequently used and have a reasonable user perceived complexity, are realised. Perceived user complexity plays a binding role, since a visually expressed domain or query construct is virtually non-existent, even counterproductive, if end users are not able to comprehend and use it.

OptiqueVQS relies on passive exploration. The interface communicates the following ontology constructs visually to aid users to reach an understanding of the underlying domain of interest: classes, class hierarchy,

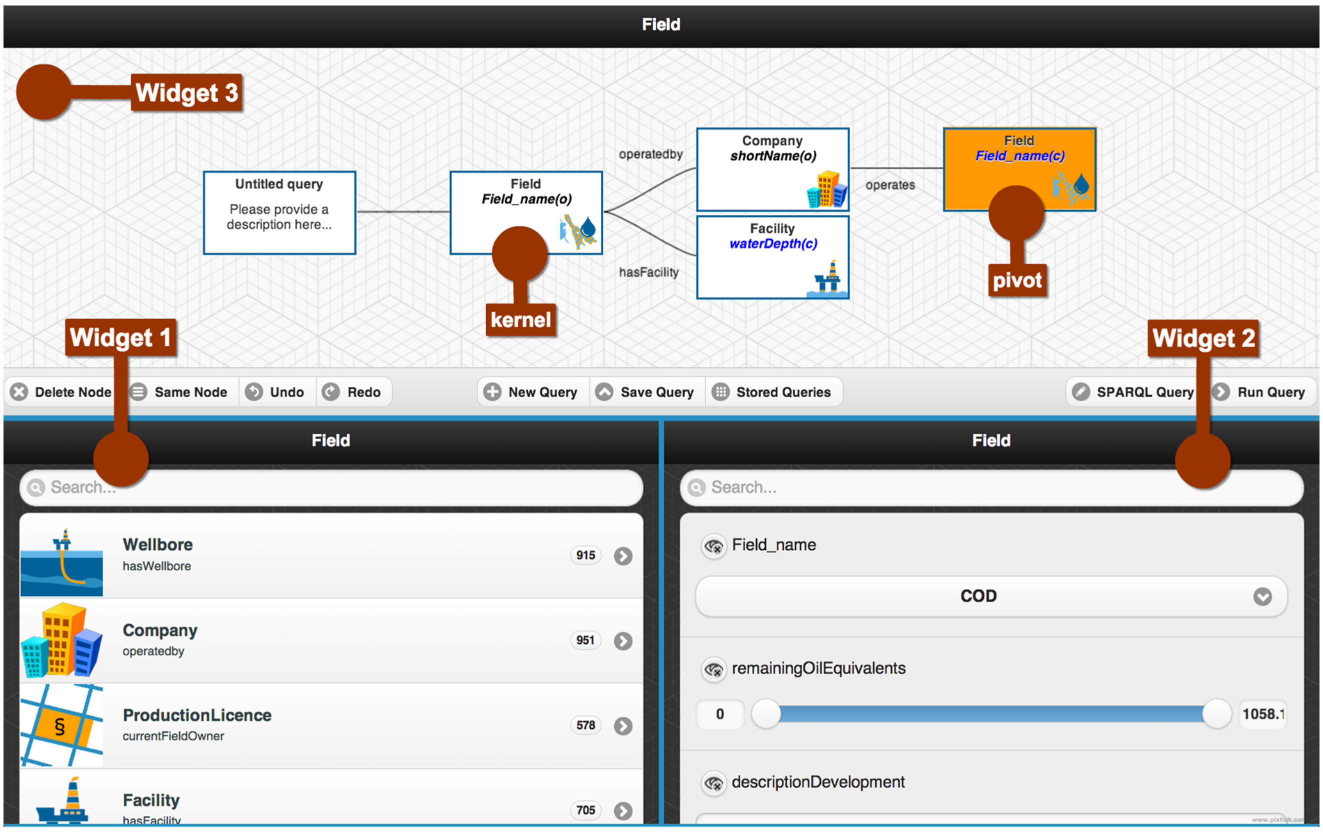


Fig. 3 OptiqueVQS – an example query is depicted for the Statoil use case.

object type properties, data type properties, multiple inheritance, enumerated classes, inverse relationships, and multiple ranges. Visual representations of disjointness and subproperties are also of use; however, the interface does not accommodate them currently. The interface also supports the propagation of property restrictions.

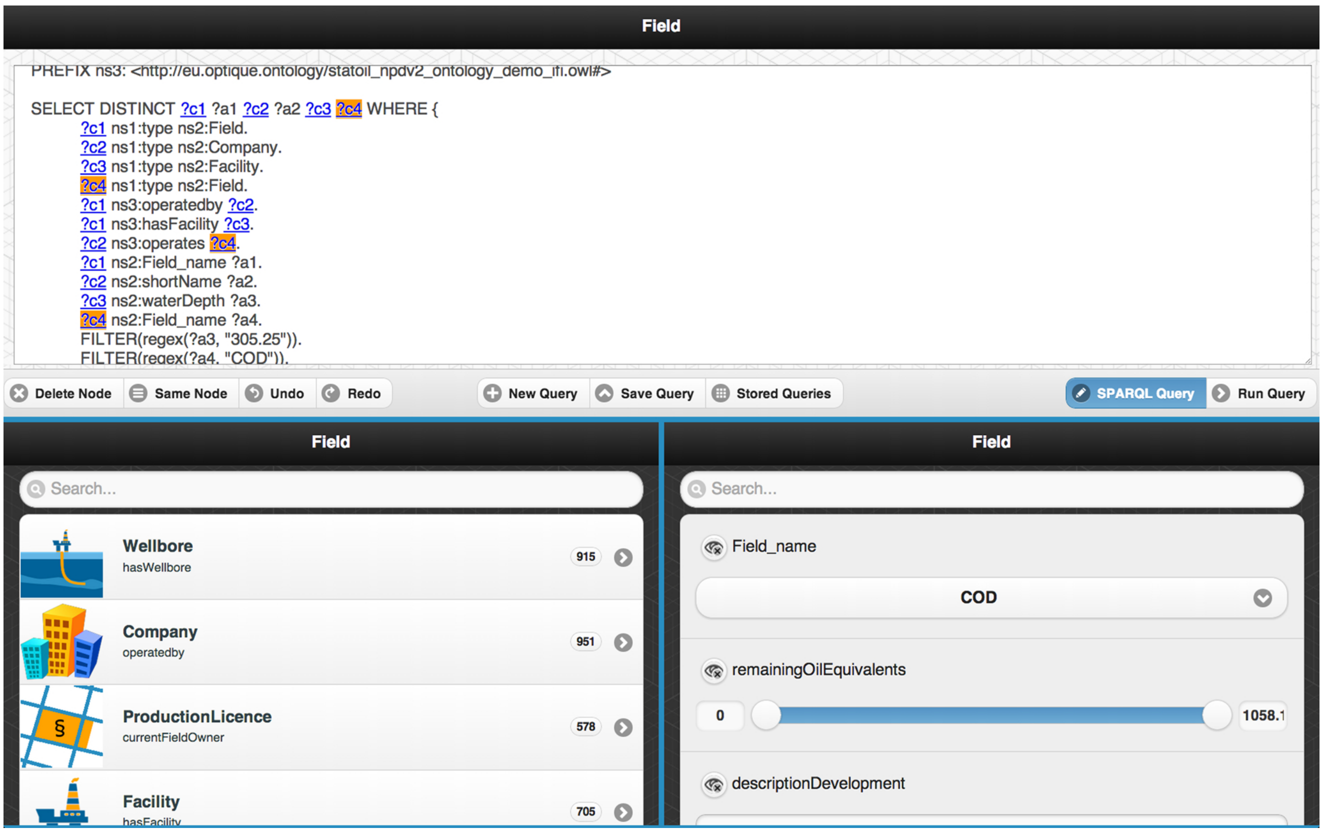
The interface allows construction of linear and tree-shaped conjunctive queries, i.e., first level, as this is set as the ground challenge (cf. Section 4). As far as the second level is concerned, three types of cycles that matter for end users at the interface level are identified; *type I cycles* are formed when a variable-node is referred more than once, for instance, “give me all the people who live and work in the same city”. *Type II cycles* occur when there exist at least one comparison between the attributes of two distinct variable-nodes, for instance, “give me all the people whose fathers are older than their mothers”. Lastly, *Type III cycles* occur when there exists at least two variable-nodes of the same type. Cycles of type III are supported naturally, since at the interface level every variable-node is implicitly typed rather than having concept-nodes for typing. Type I cycles are expected to be supported through node duplication, where end users can assert that two variables are indeed the same (cf. “Same Node” button in Figure 3); as already stated, this is to keep queries in tree-shape for the usability purposes (cf. [16]). Queries that involve disjunction and aggregation are not accommodated by the interface. Concerning the third level of queries, existential quantification is implicitly supported through the interface; however, queries that include universal quantification and negation are not supported.

Note that support for disjunctions, universal quantifiers, and negation is also a concern for the underlying OBDA framework, which is introduced in Section 6.

5.3 Design rationale

A VQS, rather than a formal VQL, has been intentionally chosen, since a VQS typically employs ad-hoc representations and affordances and relies on common knowledge and skills rather than users’ ability to learn a new language and syntax (cf. [74,27]). This allows to avoid rigid boundaries of a formal language and provides a good basis for usability and expressivity balance.

In order to address the usability and expressiveness trade-off, as stated previously, the focus is on user perceived complexity and the frequency of use – the former being the predominant decision criteria. End users are



```
PREFIX ns3: <http://eu.optique.ontology/statoil_npdv2_ontology_demo_ifi.owl#>

SELECT DISTINCT ?c1 ?a1 ?c2 ?a2 ?c3 ?c4 WHERE {
        ?c1 ns1:type ns2:Field.
        ?c2 ns1:type ns2:Company.
        ?c3 ns1:type ns2:Facility.
        ?c4 ns1:type ns2:Field.
        ?c1 ns3:operatedby ?c2.
        ?c1 ns3:hasFacility ?c3.
        ?c2 ns3:operates ?c4.
        ?c1 ns2:Field_name ?a1.
        ?c2 ns2:shortName ?a2.
        ?c3 ns2:waterDepth ?a3.
        ?c4 ns2:Field_name ?a4.
        FILTER(regex(?a3, "305.25")).
        FILTER(regex(?a4, "COD")).
```

Fig. 4 OptiqueVQS – the example query is depicted in SPARQL text view.

also enabled to modify existing queries to fit them to the task at hand and to formulate more complex queries. Query reuse is a *passive collaboration* technique (cf. [60]), which saves user effort and time and has a didactic role, in terms of knowledge share through examples, for the training of end users.

OptiqueVQS provides an instant, reversible and iterative experience and addresses the scalability issue against large ontologies by gradually loading on demand the information about classes, and offering search fields to find classes and properties without having to navigate endless lists.

The visual representation and interaction paradigms along with underlying metaphors, analogies, etc. are of primary importance for a VQS. A single representation and interaction paradigm are not sufficient for addressing main data access activities, i.e., exploration and construction, at an acceptable level of expressiveness and usability. Therefore, OptiqueVQS combines different paradigms and the best practices to reach its goal succesfully. The architectural choice of OptiqueVQS, which is described in Section 6, plays a crucial role in this respect; the mashup approach built on widgets underpins the flexibility and extensibility of OptiqueVQS, so that one can combine different representation and interaction styles (i.e., multi-paradigm). In this respect, the following address each widget on an individual basis.

W1 follows a list/menu-based representation style and is capable of presenting a considerably high number of items, along with supportive icons and meta information. Navigational interaction employed by W1 is a familiar paradigm for end users as stated earlier (cf. [87]). In W1, each selection is a combination of a relation with its range concept; this helps to reduce the number of navigational levels that the user has to pass through (cf. [79]).

In W2, the form-based representation style and range selection interaction style are employed; these paradigms are well known by end users and known to be intuitive as well (cf. [17]). A limited amount of faceted search flavour is provided in W2, since frequent database access is not feasible in the present context (i.e., due to the large data size). W2 realises subclass refinement through a flat list, rather than a tree-based structure. Although tree-based structures are commonly employed for visualising hierarchies, trees immediately become problematic for large hierarchies (cf. [47]).

Last but not the least, in W3, a diagram-based approach is used; diagrams are good at communicating relationships over a spatial dimension (cf. [47]). In W3, queries are represented in the form of trees, including the cyclic queries (cf. [16]), since graphs representing information needs could easily get complex and hard to comprehend, due to cycles, and lack of a kernel concept and query flow direction. One could consider the query visualisation provided by W3 in terms of a VQL; however, it is rather ad-hoc and informal.

Holistically, W3 is meant to provide a constant overview of the active query task, while W1 and W2 are meant to keep the focus on the pivot concept and to enable users to iteratively formulate their queries. In this way, OptiqueVQS provides a clear distinction and support for view and overview. Furthermore, each widget employs human readable labels of ontology elements rather than using their identifiers.

6 Implementation and architecture

A prototype was implemented as a proof of concept; an extensive demo video is available online[7]. The client side is based on the *HTML 5*, *JavaScript*, and *CSS*, more specifically, on *jQuery* (a *cross-platform* Javascript library), *jQuery Mobile* (a mobile web framework built on jQuery), and *InfoVis* (a visualisation library built on JavaScript). A mobile web framework was particularly selected for the implementation, since the fact that it tailors the look and feel for small screens and provides a touch optimised experience enables *cross-device* experiences (e.g., the use of OptiqueVQS or any of its widgets in the field through mobile devices).

The backend is built on the *Information Workbench*[8] (IWB) [38]. IWB is a generic platform for semantic data management, which provides a shared triple store for managing the OBDA system assets (such as ontologies, mappings, query logs, (excerpts of) query answers, database metadata, etc.), generic interfaces, APIs for semantic data management (e.g., ontology processing APIs), and other end-user related components, such as for result visualisation and textual query editing. IWB does not only host the widget backends, but also integrates all the components and APIs developed by the project partners (cf. Section 2), such as *ontop*[9] [69,70] for query rewriting and optimisation and mapping management, and *ADP* [52] for distributed query processing. A special API was also developed to allow widgets to access and query the triple store and ontologies.

The OBDA framework behind OptiqueVQS supports *OWL 2 QL* [62] and a conjunctive fragment of *SPARQL 1.1* [39]. In OBDA, the main concern, while selecting an ontology language, is to achieve reasonable computational complexity with respect to the size of data; it is now well known that efficiency of query answering cannot be guaranteed if the ontology is expressed in a logic whose expressive power exceeds the one of lightweight language [20]. OWL 2 QL is a *profile* of *OWL 2* and is based on the *DL-Lite* family of *description logics* (DL) (cf. [36]). OWL 2 QL is particularly meant for applications for query answering with a large amount of instance data and in this profile query answering can be implemented by rewriting queries into a standard relational query language [62]. Note that although within the scope of Optique one is restricted to OWL 2 QL, OptiqueVQS is generic and could be used with more expressive profiles of OWL.

6.1 Formal behaviour

The way the ontology controls the behaviour of OptiqueVQS should be seen from two perspectives: from a *knowledge representation* (*KR*) perspective, Optique exploits the graph-based organisation of ontological elements and data for representing the domain and query structures (cf. query by navigation); from a *logic* perspective, it uses ontological *axioms* to constrain the behaviour of the interface and to extend the available knowledge.

On a purely structural level, OptiqueVQS could be controlled directly by a graph G that captures the concepts and the properties of an ontology O. An OWL ontology can be viewed as a *labelled directed* RDF graph $G = (N, E)$, where N is a finite set of labelled *nodes* and E is a finite set of labelled *edges* (cf. [36]). A pairwise disjoint alphabets U, a set of *URIs*, L, a set of *terminal literals*, and B, a set of *blank nodes* are considered. An edge is a *triple* written in the form of $\langle s, p, o\rangle \in (U \cup B) \times U \times (U \cup L \cup B)$. The nodes of the graph mainly represent concepts and edges represent properties. A SPARQL query is formally represented by a tuple defined as $Q = (A, V, D, P, M, R)$. A is the set of *prefix declarations*, V is the *output form*, D is the *RDF graph* being queried, P is a *graph pattern*, M are *query modifiers*, which allow to modify the results by applying *projection*, *order*, *limit*, and *offset* options. SPARQL is based on matching graph patterns against RDF graphs. P is composed of a set of *triple patterns* and describes a *subgraph* of D. The main difference between a triple pattern and RDF triple comes from the fact that the former may have each of *subject*, *predicate* and *object* as a variable. However, once variables in triple patterns are substituted with constants or blank nodes, an RDF graph $P'(N', E')$, which could be considered as a subgraph of the actual RDF data graph, is obtained.

[7] `http://youtu.be/ks5tcPZVHp0`

[8] `www.fluidops.com/information-workbench/`

[9] `http://ontop.inf.unibz.it`

Every query generated by OptiqueVQS has a graph pattern represented by a set of triple patterns, where each triple pattern is a tuple $t \in Var \times U \times (U \cup Var \cup L)$ and Var is an infinite set of variables. An example SPARQL graph pattern is depicted in Figure 5 for the query example presented in Figure 3 and Figure 4. The state of an edited query is composed of a partial graph pattern and a *cursor position* (cf. pivot). The cursor position is either blank (i.e., empty query) or points to a variable in the query. If the query is empty, the selection of a concept v from W2 results in a new tuple $\langle x, \texttt{rdf:type}, v\rangle \in Var \times U \times U$ in P, where x is a fresh variable. If the cursor points to a variable x, of type v, then each selection of an object property o with target class w from W1 (corresponding to an edge $\langle v, o, w\rangle \in G$) adds the following two triple patterns to P: $\langle x, o, y\rangle \in Var \times U \times Var$ and $\langle y, \texttt{rdf:type}, w\rangle \in Var \times U \times U$, where y is a fresh variable. Every selection and projection operation realised over a data property d in W3, while cursor is on a variable x, adds a new tuple $\langle x, d, y\rangle \in Var \times U \times (Var \cup L)$ to P. Finally, the selection of a subclass v for a typed variable x in W3 results in a new triple in P: $\langle x, \texttt{rdf:type}, v\rangle \in Var \times U \times U$.

Comparing the SPARQL graph pattern given in Figure 5 to the visual representation given in Figure 3, it can be seen that the *concept-nodes*, *variable-nodes* referring to literals, and *literal-nodes* are omitted along with the edges connecting them; and variable-nodes only represent individuals. The type information, output form, and constraints on attributes are embedded into each corresponding variable-node.

6.2 Backend support

The domain knowledge that needs to be visualised through the visual query interface is often richer than what an OWL 2 QL ontology can express. In this case the resulting ontology will fall outside the OWL 2 QL (e.g., *cardinality restrictions*); in the OBDA framework, this issue is addressed through *approximation* techniques (cf. Figure 6), which transform the ontology into one that is as expressive as possible, while still falling within the required profile (cf. [65,22,21]).

For instance, multiple property domains and ranges are not permitted in OWL 2 QL. Thus an OWL 2 QL approximator is provided [22,21] to approximate or remove such axioms. However, although this information is not used in the rewriting process, it does have an important role in the OptiqueVQS. Hence, in order to be able to keep this non OWL 2 QL information, annotations are added to the ontology about the multiple domains and ranges based on OWL 2 annotations[10].

Moreover the ontology is enriched with more annotations. These are user-friendly labels for concept and properties and annotations pre-computed from the underlying data for increasing the responsiveness of the interface. For instance, values that are frequently used and rarely changed are pre-computed; 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), which also fall outside OWL 2 QL.

The OBDA framework helps to increase the expressiveness of the interface through delegating some complex notions to the mapping layer. This layer allows to define mappings from a relational database to an RDF or OWL vocabulary of concepts/properties and let the system view the database as if it was an RDF graph. This layer also allows to abstract complex notions into new relationships and concepts, and define built-in functions. Listing 1 and Listing 2 together form a mapping (i.e., for ontop [2]) and present a simplified example from the Statoil use case.

Listing 1 Mapping target – triple template.

```
<"&:;M1-{$M1_id}">
        :intersects
<"&:;M2-{$M2_id}}"> .
```

Listing 2 Mapping source – SQL code.

```
SELECT M1.id as M1_id, M2.id as M2_id
FROM Measurement1 as M1, Measurement2 as M2
WHERE M1.id = M2.id
AND (M1.top BETWEEN M2.top AND M2.bottom
OR M1.bottom BETWEEN M2.top AND M2.bottom)
```

Two relations are assumed in a relational database, namely Measurement1 and Measurement2, which are mapped to their corresponding concepts in an ontology (omitted). In Listing 1 and Listing 2, a new complex relationship named “intersects” is defined in the ontology. Then, it is instantiated by asserting a new triple for every tuple emerging as a result of the joining of Measurement1 and Measurement2 relations on the condition that there is an overlap between their intervals defined with top and bottom attributes. The source given in Listing 2 is an arbitrary SQL query over the database and the target given in Listing 1 is a triple template containing the placeholders that reference attribute names mentioned in the source query. This way an end user can simply use the new relationship rather than comparing top and bottom intervals of two variable-nodes

[10] http://www.w3.org/TR/owl2-new-features/#Extended_Annotations

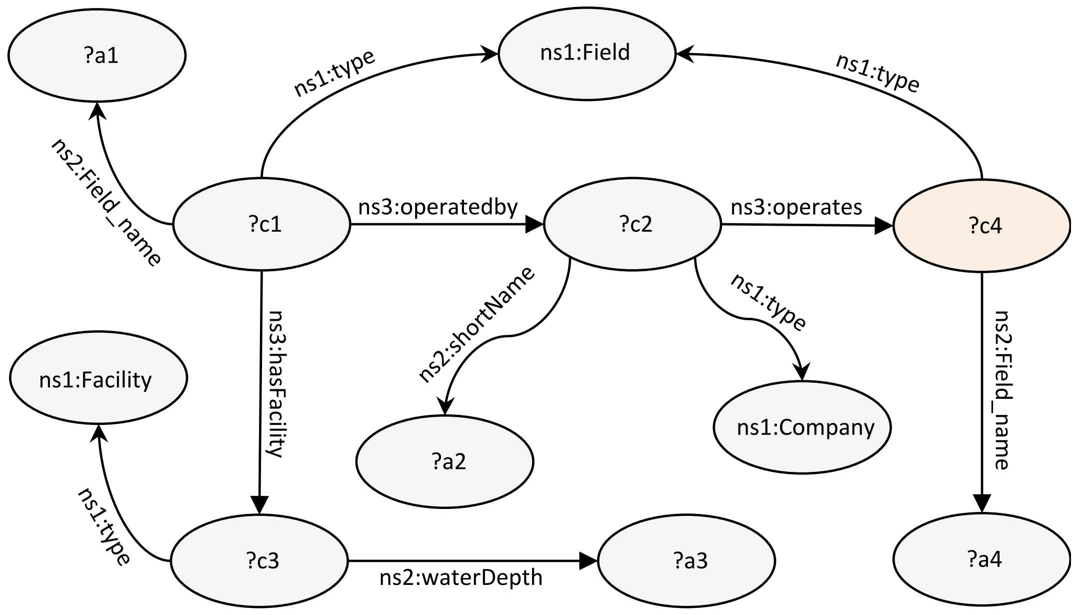


Fig. 5 An example SPARQL graph pattern is depicted for the Statoil use case.

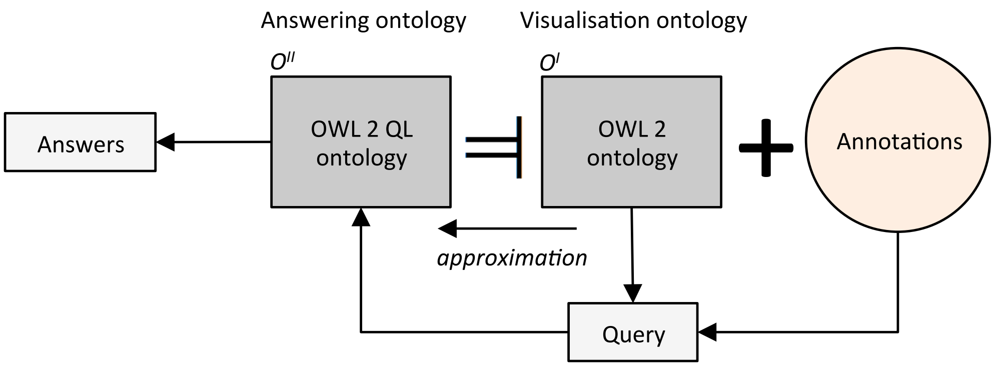


Fig. 6 Ontology approximation for query answering and visualisation.

through the interface; another similar example could be for checking whether a geographical area contains a given location – both represented by their coordinates. This approach empowers end users to formulate more complex queries and could increase the usability of the interface.

6.3 Interface architecture

Widgets are managed by a *widget runtime environment*, which provides basic *communication and persistence services* to the widgets, while the presence of widgets (location, size, etc.) on the interface is managed by *widget containers*. The *orchestration* of widgets relies on the requirement that each widget discloses its functionality to the environment through a client side interface and notifies any other widget in the environment (e.g., broadcast and subscription) and/or the widget environment upon each user action. Then, either each widget decides what action to execute in response, by considering the *syntactic* or *semantic signature* of the received event, or the environment decides on widgets to invoke. The core benefits of such an approach are as follows:

– It becomes easier to deal with complexity, since the management of functionality and data could be delegated to different widgets.
– Each widget could employ a different representation and interaction paradigm that best suits its functionality.
– Widgets could be used alone or together, in different combinations, for different contexts and experiences.
– The functionality of the overall interface could be extended by introducing new widgets.

The current architecture of OptiqueVQS is depicted in Figure 7. The architecture assumes that each widget has client side and server side components (for complex processing), and that widgets can communicate with each other and with the runtime environment through a *communication channel*. The communication channel resides at the client side and is built on *post message* method of HTML 5. Each widget also has a *data port*, which allows widgets to access server side data sources – through *REST* calls in this case. Widget runtime environment have an *environment controller* at the client side and a *component control logic* at the server side. The former is responsible for operational tasks such as collecting event notifications from widgets and submitting control commands to them. The latter is responsible for the orchestration logic, that is, it decides on which widgets should react to which events. Widgets follow the specification of the W3C [14] and the architecture is adopted from the authors' earlier work on UI mashups [80].

Note that the architecture depicted here only concerns the visual query formulation system; the overall Optique architecture, which includes other core components, such as for query evaluation, ontology management and evolution, mappings, and distributed query execution, is discussed in another publication (cf. [50]).

7 Usability evaluation

The development of OptiqueVQS follows a *user-centred design* approach (cf. [37]). Accordingly, the development of OptiqueVQS was initiated by collecting generic user requirements through observing and talking with the target users in their work context. Then, early ideas were discussed with a set of representative users through sketches illustrating the proposed design step by step; early and continuous prototypes followed soon after. An evolutionary development approach with a holistic perspective is at the core of the development cycle, that is, the development is iterative and incremental and follows a systematic approach by taking future needs and broad user context into account.

OptiqueVQS was integrated into the Optique platform and end-user workshops were organised in the premises of both uses case partners (i.e., Statoil and Siemens) in order to allow representative end users to have their first full experience with the Optique platform over their own data sources (i.e., end-to-end). After a short demo, participants, i.e., domain experts, experimented with the OptiqueVQS and formulated queries. The feedback delivered by the domain experts was positive; along with some minor requests, domain-specific components were demanded, such as maps for selecting locations.

The first experiment was carried out on a generic setting with users with no technical background, this was due to fact that such an experiment allows to test the system in a general context and to identify and fix major flaws before experimenting on domain experts, given their highly constrained availability. Note that these users are representative of the target group as they have no/very limited technical knowledge. The experiment carried out and the results obtained are presented in the following.

7.1 Experimental setting

The experiment was designed as a *think-aloud* study (cf. [30]), since the goal of the experiment was not purely *summative*, but to a large extent *formative*. The experiment is built on a "movie ontology". A visual excerpt from the vocabulary of ontology is given in Figure 8; note that inverse properties are omitted in the figure for the sake of brevity. In total, the ontology includes 6 concepts, 16 relationships (including inverse properties), and 17 attributes, which already allow to design complex queries. A larger ontology was avoided in order to omit the effect of ontology size on the query formulation in this phase.

A total of 15 participants took part in the experiment; the profiles of participants are summarised in Table 2. Participants were selected particularly among non-technical people, since they are the primary target of OptiqueVQS. 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 asks users about their *age*, *occupation* and *level of education*, and asks 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, given at most three attempts for each. An attempt is an iteration that consists of a formulation of a query and execution of it and in a subsequent attempt a user usually modifies what s/he has constructed in the previous attempt. Each participant performed the experiment in a dedicated session, while being watched by an observer. Participants were instructed to think aloud, including any difficulties they encountered (e.g., frustration and confusion), while performing the given tasks.

There were 6 tasks, representing the information needs used in the experiment (see Table 3). Each information need maps to a query at a different level

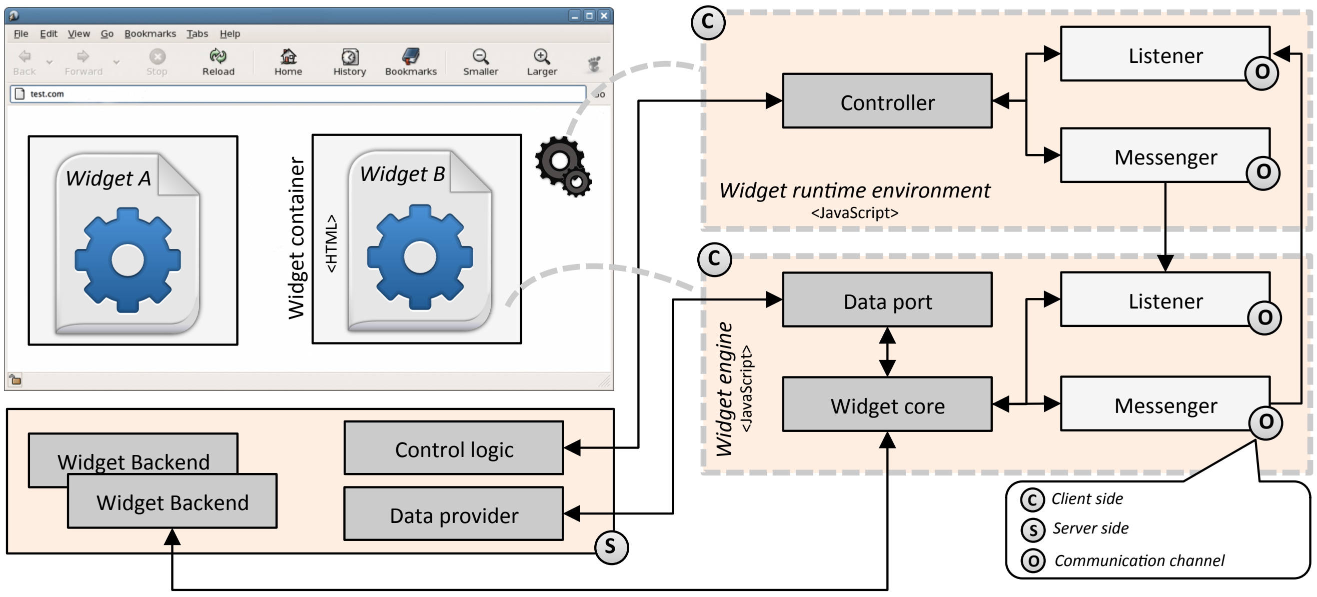


Fig. 7 OptiqueVQS architecture based on widget-based UI mashups.

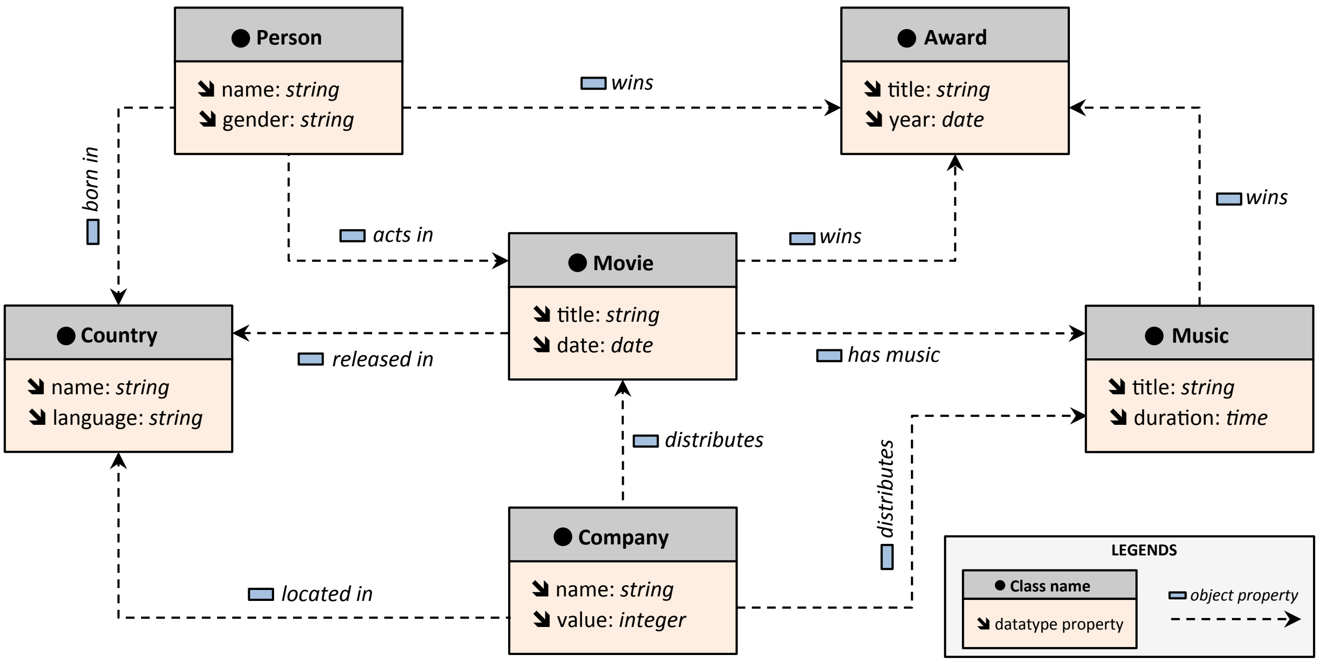


Fig. 8 A visual excerpt from the vocabulary of movie ontology used in the experiment.

of complexity with respect to its topology and length, in an increasing order of complexity (all conjunctive): *short linear (T1)*, *long linear (T2)*, *short with branching (T3)*, *long with branching (T4)*, *short with branching and type III cycle (T5)*, and *long with branching and type III cycle (T6)*. A query is classified as "long" if it has a maximum tree depth of at least 3, and as "short" otherwise.

Once users completed their tasks, they were asked to fill an exit survey concerning their experiences with the tool. The survey asks users to rate whether the questions were easy to do with the tool (*S1*), the tool was easy to learn (*S2*), easy to use (*S3*), gave a good feeling of control and awareness (*S4*), was aesthetically pleasing (*S5*), overall satisfactory (*S6*), and enjoyable to use (*S7*) on a Likert scale (i.e., 1 for "strongly disagree" and 5 for "strongly agree"). Users were also asked to comment

Table 2 Profile information of the participants.

#	Age	Occupation	Education	Technical skills	Similar tools
P1	32	Chemist	PhD	2	3
P2	26	Math teacher	Bachelor	1	1
P3	43	Law student	Master	1	1
P4	21	Political science student	Bachelor	1	2
P5	22	Criminology student	Bachelor	1	3
P6	31	Hydrology student	Master	2	4
P7	26	Complex systems student	Master	2	3
P8	23	Psychology student	Bachelor	1	3
P9	24	Finance student	Bachelor	2	3
P10	21	Law student	Bachelor	2	2
P11	21	Law student	Bachelor	1	1
P12	21	Biology student	Bachelor	1	1
P13	23	Natural sciences student	Bachelor	1	1
P14	24	History student	Bachelor	1	3
P15	22	Biology student	Bachelor	1	1
Avg.	25	-	-	1.3	2.1

Table 3 Information needs used in the experiment.

#	Query type	Information need
T1	Short linear	Find the *names* of all the *companies* that *distribute* a *movie titled* "Titanic".
T2	Long linear	Find the *names* of all the *people* who *acted in* a *movie released between* 1970 and 1980 and *distributed by* a *company located in* Germany.
T3	Short with branching	Find the *titles* of all the *musics* that *won* an *award titled* "Best of Movie Musics" and are *played in* a *movie titled* "The Red Warrior".
T4	Long with branching	Find the *titles* of all the *movies* that are *distributed by* a *company owned by* a *person born in* USA and *have* a *music* that *won award between* 1980 and 1990.
T5	Short with branching and type III cycle	Find the *names* of all the *companies* that *distribute* a *movie titled* "Titanic" and *distribute* a *music played in* a *movie released in* 1980.
T6	Long with branching and type III cycle	Find the *titles* of all the *musics distributed by* a *company located in* the UK and *played in* a *movie* that *has* an actor named "George" who was *born in a country* in the African continent and *won* an *award* in 1990.

on what they did like and dislike and to provide more feedback.

7.2 Results

The results of the experiment are presented in Table 4. A total of 90 tasks were completed by the participants with an 80 percent first-attempt correct completion rate[11] (i.e., ratio of correctly formulated queries in first attempt to the total number of tasks). On average a task needed 74 seconds to complete on 1.2 attempts; the first and fourth tasks needed the shortest and the longest times to complete, on average 34 seconds and 93 seconds respectively. The third task had the highest average in the number of attempts with 1.5, while the first and the sixth tasks had the lowest average in the number of attempts with 1 and 1.1 respectively.

According to the results and the observations, participants solved the first task (i.e., short linear) quite easily. However, when it came to the third task (short with branching), half of the participants failed in their first attempts. This is particularly due to fact that they were mostly not expecting a branching after two linear queries and did not pay attention to the text of the information need. Yet, as soon as they realised the case, they did quickly recover and manipulated their queries accordingly. The average number of attempts then decreased for the subsequent tasks (i.e., all with branching) as users became more aware. The fourth task (i.e., long with branching) needed the longest time on average, since after the third task participants paid more

[11] Lopez et al. [58] suggest using *precision*, *recall*, and *f-measure* over the returned result set; however, contrary to information retrieval, in data retrieval a query is meant to retrieve all and only those objects that match the criteria. In other words, even a single erroneous object implies a total failure (cf. [67,1]). Therefore a binary measure of success (i.e., correct/incorrect query) rather than a fuzzy one is used.

Table 4 The results of the experiment (*c* for complete, *t* for time in seconds, and *a* for attempt count).

	T1			**T2**			**T3**			**T4**			**T5**			**T6**			**Avg.**		
#	*c*	*t*	*a*	*c*	*t*	*a*	*c*	*t*	*a*	*c*	*t*	*a*	*c*	*t*	*a*	*c*	*t*	*a*	*c*	*t*	*a*
P1	1	50	1	1	70	1	1	94	2	1	55	1	1	53	1	1	68	1	1	65	1.2
P2	1	48	1	1	83	1	1	80	1	1	113	2	1	60	1	1	70	1	1	76	1.2
P3	1	81	1	1	87	1	1	80	2	1	180	2	1	141	2	1	145	1	1	119	1.5
P4	1	18	1	1	44	1	1	41	1	1	124	2	1	73	1	1	90	1	1	65	1.2
P5	1	32	1	1	85	1	1	62	1	1	74	1	1	82	1	1	85	1	1	70	1.0
P6	1	16	1	1	136	2	1	125	2	1	86	1	1	108	1	1	100	1	1	95	1.3
P7	1	27	1	1	105	2	1	102	2	1	126	2	1	122	2	1	135	1	1	103	1.7
P8	1	75	1	1	47	1	1	78	2	1	54	1	1	48	1	1	71	1	1	62	1.2
P9	1	23	1	1	59	1	1	54	1	1	82	1	1	45	1	1	81	1	1	57	1.0
P10	1	14	1	1	54	1	1	41	1	1	73	1	1	47	1	1	80	1	1	52	1.0
P11	1	17	1	1	42	1	1	65	1	1	53	1	1	105	2	1	60	1	1	57	1.2
P12	1	29	1	1	72	1	1	84	2	1	103	1	1	56	1	1	83	1	1	71	1.2
P13	1	38	1	1	54	1	1	44	1	1	75	1	1	46	1	1	80	1	1	56	1.0
P14	1	28	1	1	96	1	1	65	1	1	58	1	1	54	1	1	60	1	1	60	1.0
P15	1	19	1	1	125	2	1	112	2	1	144	1	1	50	1	1	168	2	1	103	1.5
Avg.	1	34	1	1	77	1.2	1	75	1.5	1	93	1.3	1	72	1.2	1	91	1.1	1	74	1.2

attention to clearly understand the information need. Participants solved the fifth task (i.e., short with branching and type III cycle) comparatively quickly; this was due to the short length of the query and due to the fact that participants did not have any confusion, when a concept appeared twice in the query (only one participant had this confusion and raised it). Finally, participants solved the last task (i.e., long with branching and type III cycle) quite smoothly and with confidence, although it was the longest and the most complex one (i.e., with two branches and one type III cycle). A snapshot from the final query is given in Figure 9.

The feedback provided by the participants through the exit survey is presented in Table 5 and Table 6. Participants overall rated the tool as "good" with 4 out of 5 on average. The first statement (cf. S1 – the questions were easy to do with the tool) had the lowest rank with 3.7; according to the observations, this was mostly due to the texts of the information needs, rather than the tool. The texts describing the information needs (cf. Table 3) include a number of relative pronouns along with a passive sentence structure, which make them hard to understand at a first glance and to keep in the short-term memory. Although, this structure was intentionally selected in order to avoid a step-by-step question form, for subsequent evaluations, a different form could be considered. As listed in Table 6, participants mainly found the tool orderly. Participants liked the way that queries were visualised, i.e., a diagrammatic overview. They also appreciated the fact that the tool allows them to formulate detailed information needs easily and in an organised way. The introduction given to the users was only around five minutes with an example query, therefore participants were mostly expected to learn on the way, since one of the goals was to have a tool with a low learning curve and effort. This case is reflected and confirmed by the participants' comments.

Observing the participants in action allowed to acquire some specific insights about the tool. One major issue was that while formulating the fourth task, participants initially looked for a "birth place" field in W2, since the information need was specifying a person born in the USA. It took only a while for participants to realise that this information is only accessible through a relationship rather than an attribute. A participant first considered the branches as "OR" rather than "AND" and asked whether it was possible to construct "OR" branches. Two participants realised that indeed they do not have to follow the logical order given in the descriptions of information needs (i.e., to join the concepts in the given order), and the alternatives exist. One of these participants solved one of the tasks successfully with an alternative order. Finally, from a general perspective, users did not have any major difficulties in using and learning the tool and were quick in realising the given tasks. After the experiment, many of the participants raised their interest regarding the tool, asked further questions, mostly concerning the context in which the tool is going to be used, and stated that their experience with the tool was comparable to the games in terms of the joy they had.

8 Discussion

The results suggest that OptiqueVQS is a promising tool for end-user visual query formulation with high effectiveness and efficiency. The tool provides a decent

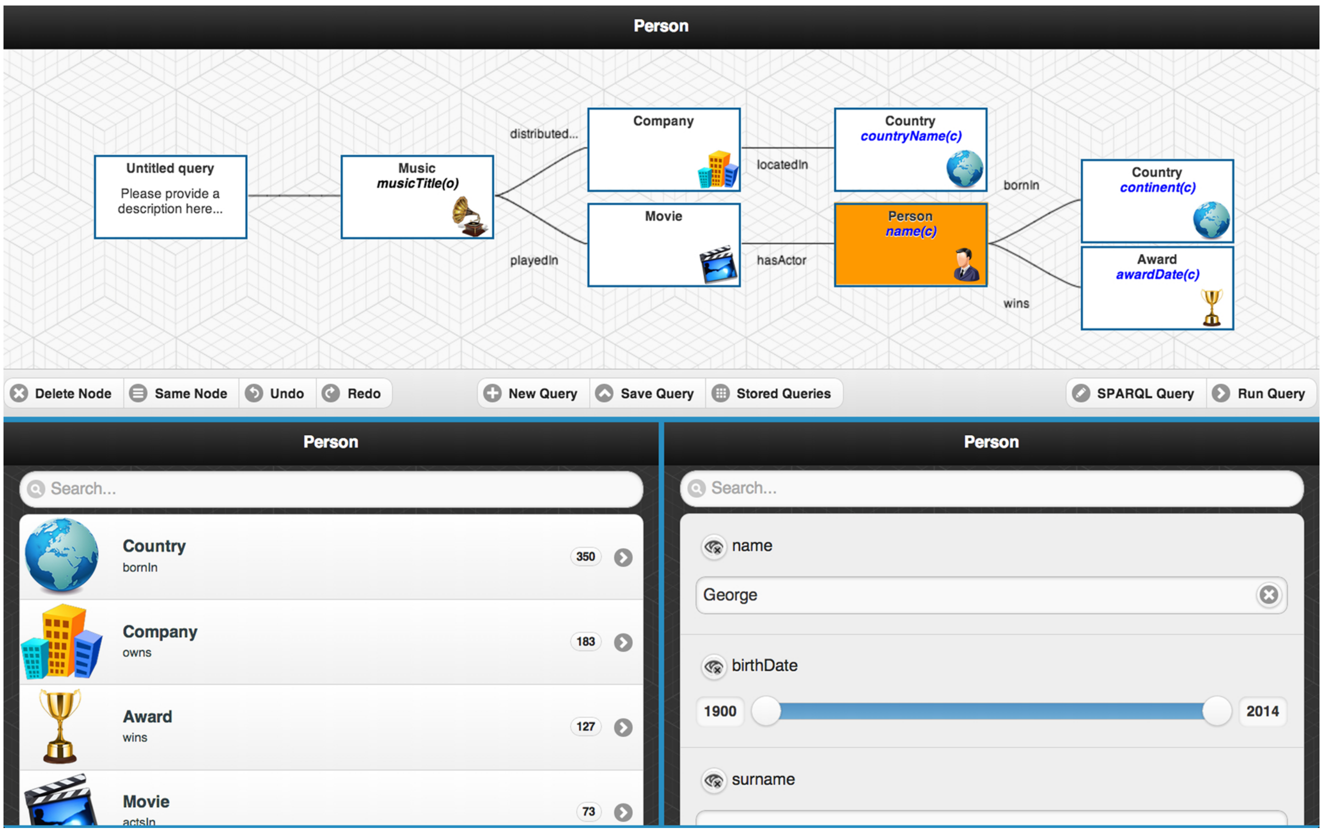


Fig. 9 An excerpt from a query formulated by the participants during the experiment.

Table 5 The results of the exit survey.

#	S1	S2	S3	S4	S5	S6	S7	Avg.
P1	5	5	5	5	4	4	5	4.7
P2	4	4	5	5	5	4	5	4.6
P3	4	4	4	4	4	4	4	4.0
P4	3	4	2	3	4	4	4	3.4
P5	4	4	4	4	4	4	5	4.1
P6	4	5	4	5	5	4	4	4.4
P7	3	3	4	4	4	4	4	3.7
P8	5	5	5	5	4	5	5	4.9
P9	3	4	4	4	4	4	4	3.9
P10	3	3	4	4	3	4	4	3.6
P11	3	4	4	4	4	4	4	3.9
P12	4	4	4	4	4	4	4	4.0
P13	4	3	4	4	4	4	4	3.9
P14	3	3	4	3	4	4	4	3.6
P15	4	4	3	3	4	4	4	3.7
Avg.	3.7	3.9	4.0	4.1	4.1	4.1	4.3	4.0

level of expressivity and demands only common user knowledge and skills, hence has a high accessibility and low learning curve. The authors' experiences both with the domain experts of the presented use case partners, as well as with end users without almost any technical skills reveal that the tool has an engaging nature and provides a good level of user satisfaction. OptiqueVQS could be used with different OBDA frameworks, independent of an OBDA framework directly on SPARQL endpoints, or on different data models and structures, such as relational models, due to its compliance with graph-based structures. Having said these, the following discuss the most prominent limitations and potential extensions and the plan to address them.

Table 6 The feedback given by the participants.

#	**Like**	**Dislike**
P1	"Visual, easy to use, fast and easy to correct mistakes."	"...can be visually improved."
P2	"Easy to jump on [diagram] and suggestions [of the W1] were relevant"	-
P3	"Easy and organised. Good for an organised and focused search."	"Nothing"
P4	"I like that it can make search process go faster and make it more specific"	"It could get complicated as you have to link and sometimes go back to previous boxes."
P5	"It was OK to find what the tasks asked for without having to look too long for the right variables."	-
P6	"The schematic diagram"	"It has fixed options."
P7	"Good overview"	-
P8	"The way you connect the nodes, the way it was easy to incorporate a lot of information in the right way, and it was easy to be organised."	"Maybe seems a bit simple at the first glance, but then it was good!"
P9	"Nice visualisation [for diagram]"	"Many steps"
P10	"Easy to use"	-
P11	"It was quite simple."	"It felt I did not have much time [to learn]."
P12	"The organisation in images and scheme"	-
P13	"The scheme on top is pretty helpful to see where you are actually getting to what you are looking for"	"It took me some time to get used to it, but then I think it works!"
P14	"You could really go in to details and ask many things about same person/company etc."	"It was a bit tricky to learn, but I think that it is possible to get a hang on it if you use it for a while"
P15	"Easy access for specific information regarding the search options: movies, music etc."	"Some difficulties [for] managing the correct search option"

The very first concern is to increase the expressiveness of OptiqueVQS, without compromising its usability. This involves extending the expressivity of OptiqueVQS with new representations and affordances, and the introduction of alternative means for the existing ones. Concerning the former, the categorisation and discussion given in Section 4 provide a systematic perspective. Therefore, the authors intend to extend the expressivity of OptiqueVQS towards level II. At this point the plan is to avoid addressing negation and universal quantifiers and leave them completely to IT experts through a textual editor. Generally, the benefit gained by incorporating rarely used complex query constructs does not make up for the loss in usability. Concerning the latter, a particular example is the one observed during the user study, that is attribute and relationship controversy. A convenient solution would be to use every relationship also as an attribute in W2, while forming its value space from a representative attribute of its range concept. For instance, take the "born in" relationship as an example in Task 4, a user can directly select the appropriate country from a "Birth place" attribute derived from the "born in" relationship, in case no other criteria are given for the country of birth; otherwise, the user can follow the "born in" relationship and impose the constraints on the "country" type variable-node.

As far as usability is concerned, *scalability* is an important challenge. Firstly, a large instance set makes the extraction of meta information, such as number of possible results upon each possible path selection (cf. W1), a challenging issue. At any point of a query task, the interface needs to execute a set of partial queries, each corresponding to the current partial query with a possible new path presented in W1. This is, in many cases, not feasible online within a reasonable time, and requires a pre-computation to derive relevant statistics. Secondly a large ontology could easily make the interface *overcrowded* and *cluttered*, and in turn could hinder users' ability to find the ontology elements that he/she is interested in (cf. [47]). Furthermore, even for small ontologies, constraint propagation, as discussed in Section 4, increases the number of possible properties for each concept. Although OptiqueVQS provides a gradual access to domain knowledge, this could not be a solution alone for such cases. Approaches that filter out unnecessary domain knowledge at every stage of interaction are necessary in order to ensure successful user experiences against large ontologies. This fact points to another research question, which concerns the capability of a system to adaptively suggest concepts, relationships, and attributes. Interested readers can refer to the authors' ongoing work on adaptive visual query formulation for OptiqueVQS, which exploits the query history of users to rank and suggest ontology elements with respect to an incomplete query that a user has constructed so far [78].

A visual query formulation tool should be dynamically tailored to the *context* (cf. [23]), such as *personal*, *data-related*, *task-related*, *hardware-related*, *organisational*, and *environmental*. This, on the one hand, includes automatic alterations of an application behaviour and presentation with respect to context (i.e., *adaptivity*), and on other includes user-managed customisations of the application for adapting, extending, and enriching its functionality and presentation (i.e., *adaptability*) (cf. [11]). Adaptivity and adaptability will not only help in tackling scalability issues, but will also help to provide improved user experiences. As discussed, a notable example is to find, rank, and suggest only the most relevant relationships, paths, and attributes at any point with respect to query logs (cf. [51]), and a set of heuristics defined over data (e.g., number of related instances between two type of concepts) and ontology (e.g., number of incoming and outgoing links for a concept). The widget-based architecture has a primary role in the adaptability of the OptiqueVQS. For instance, it enables end users to distribute widgets into multiple-screens, in order to open up a larger visual working space, and with an ubiquitous communication support users could remotely collaborate on the same query (i.e., distributed user interfaces – cf. [32]).

One final concern is the manual effort required for the annotation of large ontologies with human readable labels, descriptions, and icons. OptiqueVQS uses annotations when provided and identifiers otherwise; however, a sustainable solution would involve an initial set of annotations for the core ontology elements along with a mechanism to enable end-users to annotate ontology elements through the interface over time. Such an approach considers an ontology as an outcome/artefact of the work process, rather than a mere input, and is a valuable practice, since it adds end users to the loop.

9 Conclusion

This article has presented a novel multi-paradigm and ontology-based VQS, named OptiqueVQS, for end users with no technical knowledge and skills. It is built on a powerful OBDA framework and has a flexible and extensible architecture allowing to combine and orchestrate different representation and interaction paradigms. The results of the usability experiment suggest that OptiqueVQS provides a decent level of expressivity and high usability.

Finally a set of future research challenges, which includes a higher level of expressiveness, scalability, adaptivity, and annotation management, were elaborated. The systematic design and architecture of OptiqueVQS offer a sufficient and convenient room to accommodate these next level challenges. Further usability studies are planned with larger user groups and in real scenarios provided by the industrial use case partners.

Acknowledgements This research is funded by the Seventh Framework Program (FP7) of the European Commission under Grant Agreement 318338, "Optique". Ian Horroks and Ernesto Jimenez-Ruiz were also supported by the EPSRC projects MaSI^3, Score! and DBOnto.

References

1. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley (1999)
2. Bagosi, T., Calvanese, D., Hardi, J., Komla-Ebri, S., Lanti, D., Rezk, M., Rodriguez-Muro, M., Slusnys, M., Xiao, G.: The Ontop Framework for Ontology Based Data Access. In: Proceedings of the 8th Chinese Semantic Web Symposium and Web Science Conference (CSWS 2014), *CCIS*, vol. 480, pp. 67–77. Springer-Verlag (2014). DOI 10.1007/978-3-662-45495-4_6
3. Barzdins, G., Liepins, E., Veilande, M., Zviedris, M.: 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), *Frontiers in Artificial Intelligence and Applications*, vol. 187, pp. 105–116. IOS Press (2009). DOI 10.3233/978-1-58603-939-4-105
4. Bechhofer, S., Stevens, R., Ng, G., Jacoby, A., Goble, C.: Guiding the user: an ontology driven interface. In: Proceedings of the User Interfaces to Data Intensive Systems, pp. 158–161. IEEE Computer Society (1999). DOI 10.1109/UIDIS.1999.791472
5. Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., Sheets, D.: Tabulator: Exploring and Analyzing linked data on the Semantic Web. In: Proceedings of the 3rd International Semantic Web User Interaction Workshop (SWUI 2006) (2006)
6. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web - a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American **284**(5), 34–43 (2001)
7. Bevan, N., Macleod, M.: Usability measurement in context. Behaviour and Information Technology **13**(1-2), 132–145 (1994). DOI 10.1080/01449299408914592
8. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. International Journal on Semantic Web and Information Systems **5**(3), 1–22 (2009). DOI 10.4018/jswis.2009081901
9. Bobed, C., Esteban, G., Mena, E.: 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). DOI 10.3233/KES-130255

10. Brunk, S., Heim, P.: tFacet: Hierarchical Faceted Exploration of Semantic Data Using Well-Known Interaction Concepts. In: Proceedings of the International Workshop on Data-Centric Interactions on the Web (DCI 2011), *CEUR Workshop Proceedings*, vol. 817, pp. 31–36. CEUR-WS.org (2011)
11. Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.): The adaptive web: methods and strategies of web personalization. Springer-Verlag, Berlin, Heidelberg (2007)
12. Burnett, M.M.: Visual Programming. In: J.G. Webster (ed.) Wiley Encyclopedia of Electrical and Electronics Engineering. John Wiley & Sons Inc. (1999). DOI 10.1002/047134608X.W1707
13. Burnett, M.M., Baker, M.J.: A Classification System for Visual Programming Languages. Journal of Visual Languages and Computing **5**(3), 287–300 (1994). DOI 10.1006/jvlc.1994.1015
14. Cáceres, M.: Packaged Web Apps (Widgets) - Packaging and XML Configuration (Second Edition). W3C Recommendation, W3C (2012). URL `http://www.w3.org/TR/widgets/`
15. Campinas, S., Perry, T.E., Ceccarelli, D., Delbru, R., Tummarello, G.: 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), pp. 261–266. IEEE Computer Society (2012). DOI 10.1109/DEXA.2012.38
16. Catarci, T.: What happened when database researchers met usability. Information Systems **25**(3), 177–212 (2000). DOI 10.1016/S0306-4379(00)00015-6
17. Catarci, T., Costabile, M.F., Levialdi, S., Batini, C.: Visual query systems for databases: A survey. Journal of Visual Languages and Computing **8**(2), 215–260 (1997). DOI 10.1006/jvlc.1997.0037
18. Catarci, T., Dongilli, P., Di Mascio, T., Franconi, E., Santucci, G., Tessaris, S.: An ontology based visual tool for query formulation support. In: Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004), *Frontiers in Artificial Intelligence and Applications*, vol. 110, pp. 308–312. IOS Press (2004)
19. Claussen, J., Kemper, A., Moerkotte, G., Peithner, K., Steinbrunn, M.: Optimization and evaluation of disjunctive queries. IEEE Transactions on Knowledge and Data Engineering **12**(2), 238–260 (2000). DOI 10.1109/69.842265
20. Console, M., Lenzerini, M., Mancini, R., Rosati, R., Ruzzi, M.: Synthesizing Extensional Constraints in Ontology-Based Data Access. In: Proceedings of the 26th International Workshop on Description Logics (DL 2013), *CEUR Workshop Proceedings*, vol. 1014, pp. 628–639. CEUR-WS.org (2013)
21. Console, M., Mora, J., Rosati, R., Santarelli, V., Fabio Savo, D.: Effective Computation of Maximal Sound Approximations of Description Logic Ontologies. In: Proceedings of the 13th International Semantic Web Conference (ISWC 2014), *LNCS*, vol. 8797, pp. 164–179. Springer-Verlag (2014). DOI 10.1007/978-3-319-11915-1_11
22. Console, M., Santarelli, V., Savo, D.F.: Efficient Approximation in DL-Lite of OWL 2 Ontologies. In: Proceedings of the 26th International Workshop on Description Logics (DL 2013), *CEUR Workshop Proceedings*, vol. 1014, pp. 132–143. CEUR-WS.org (2013)
23. Coutaz, J., Crowley, J.L., Dobson, S., Garlan, D.: Context is key. Communications of the ACM **48**(3), 49–53 (2005). DOI 10.1145/1047671.1047703
24. Crompton, J.: Keynote talk, the W3C Workshop on Semantic Web in Oil & Gas Industry: Houston, TX, USA, 9–10 December (2008). Available from http://www.w3.org/2008/12/ogws-slides/Crompton.pdf
25. Damljanovic, D., Agatonovic, M., Cunningham, H., Bontcheva, K.: 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). DOI 10.1016/j.websem.2013.02.002
26. Dowse, R., Ehlers, M.: Medicine labels incorporating pictograms: do they influence understanding and adherence? Patient Education and Counseling **58**, 63–70 (2005). DOI 10.1016/j.pec.2004.06.012
27. Epstein, R.G.: The TableTalk Query Language. Journal of Visual Languages and Computing **2**(2), 115–141 (1991). DOI 10.1016/S1045-926X(05)80026-6
28. Erwig, M.: Xing: a visual XML query language. Journal of Visual Languages and Computing **14**(1), 5–45 (2003). DOI 10.1016/S1045-926X(02)00074-5
29. Fadhil, A., Haarslev, V.: GLOO: A Graphical Query Language for OWL Ontologies. In: Proceedings of the OWL: Experiences and Directions (OWLED 2006), *CEUR Workshop Proceedings*, vol. 216. CEUR-WS.org (2006)
30. Ferre, X., Juristo, N., Windl, H., Constantine, L.: Usability basics for software developers. IEEE Software **18**(1), 22–29 (2001). DOI 10.1109/52.903160
31. Gaines, B.R.: Designing Visual Languages for Description Logics. Journal of Logic, Language and Information **18**(2), 217–250 (2009). DOI 10.1007/s10849-008-9078-1
32. Gallud, J.A., Lozano, M.D., Vanderdonckt, J.: Distributed user interfaces: Usability and collaboration. International Journal of Human-Computer Studies **72**(1), 44 (2014). DOI http://dx.doi.org/10.1016/j.ijhcs.2013.10.006
33. Giese, M., Calvanese, D., Horrocks, I., Ioannidis, Y., Klappi, H., Koubarakis, M., Lenzerini, M., Moller, R., Ozcep, O., Rodriguez Muro, M., Rosati, R., Schlatte, R., Soylu, A., Waaler, A.: Scalable End-user Access to Big Data. In: A. Rajendra (ed.) Big Data Computing. Chapman and Hall/CRC (2013)
34. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic SHIQ. Journal of Artificial Intelligence Research **31**(1), 157–204 (2008)
35. Grau, B.C., Giese, M., Horrocks, I., Hubauer, T., Jimenez-Ruiz, E., Kharlamov, E., Schmidt, M., Soylu, A., Zheleznyakov, D.: Towards Query Formulation and Query-Driven Ontology Extensions in OBDA Systems. In: Proceedings of 10th OWL: Experiences and Directions Workshop (OWLED 2013), *CEUR Workshop Proceedings*, vol. 1080. CEUR-WS.org (2013)
36. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The Next Step for OWL. Web Semantics: Science, Services and Agents on the World Wide Web **6**(4), 309–322 (2008). DOI 10.1016/j.websem.2008.05.001
37. Gulliksen, J., Goransson, B., Boivie, I., Blomkvist, S., Persson, J., Cajander, A.: Key principles for user-centred systems design. Behaviour & Information Technology **22**(6), 397–409 (2003). DOI 10.1080/01449290310001624329
38. Haase, P., Schmidt, M., Schwarte, A.: The Information Workbench as a Self-Service Platform for Linked Data Applications. In: Proceedings of the 2nd International Workshop on Consuming Linked Data (COLD 2011), *CEUR Workshop Proceedings*, vol. 782. CEUR-WS.org (2011)
39. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language. W3C Recommendation, W3C (2013). URL `http://www.w3.org/TR/sparql11-query/`

40. Harth, A.: VisiNav: A system for visual search and navigation on web data. Web Semantics: Science, Services and Agents on the World Wide Web **8**(4), 348–354 (2010). DOI 10.1016/j.websem.2010.08.001
41. Heim, P., Ziegler, J.: Faceted visual exploration of semantic data. In: Proceedings of the 2nd IFIP WG 13.7 conference on Human-computer interaction and visualization (HCIV 2009), *LNCS*, vol. 6431, pp. 58–75. Springer-Verlag (2011). DOI 10.1007/978-3-642-19641-6_5
42. Henderson-Sellers, B.: Bridging metamodels and ontologies in software engineering. Journal of Systems and Software **84**(2), 301–313 (2011). DOI 10.1016/j.jss.2010.10.025
43. Hogenboom, F., Milea, V., Frasincar, F., Kaymak, U.: RDF-GL: A SPARQL-Based Graphical Query Language for RDF. In: R. Chbeir, Y. Badr, A. Abraham, A.E. Hassanien (eds.) Emergent Web Intelligence: Advanced Information Retrieval, Advanced Information and Knowledge Processing, pp. 87–116. Springer-Verlag (2010). DOI 10.1007/978-1-84996-074-8_4
44. Holcomb, P.J., Grainger, J.: On the Time Course of Visual Word Recognition: An Event-related Potential Investigation using Masked Repetition Priming. Journal of Cognitive Neuroscience **18**(10), 1631–1643 (2006)
45. Huynh, D.F., Karger, D.R.: Parallax and companion: set-based browsing for the data web. available online (2009). URL `http://davidhuynh.net/media/papers/2009/www2009-parallax.pdf`
46. Huynh, D.F., Karger, D.R., Miller, R.C.: Exhibit: lightweight structured data publishing. In: Proceedings of the 16th International Conference on World Wide Web (WWW 2007), pp. 737–746. ACM (2007). DOI 10.1145/1242572.1242672
47. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E.: Ontology visualization methods - A survey. ACM Computing Surveys **39**(4), 10:1–10:43 (2007). DOI 10.1145/1287620.1287621
48. Kaufmann, E., Bernstein, A.: 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). DOI 10.1016/j.websem.2010.06.001
49. Kawash, J.: Complex Quantification in Structured Query Language (SQL): A Tutorial Using Relational Calculus. Journal of Computers in Mathematics and Science Teaching **23**(2), 169–190 (2004)
50. Kharlamov, E., Jiménez-Ruiz, E., Zheleznyakov, D., Bilidas, D., Giese, M., Haase, P., Horrocks, I., Kllapi, H., Koubarakis, M., Özçep, O., Rodríguez-Muro, M., Rosati, R., Schmidt, M., Schlatte, R., Soylu, A., Waaler, A.: Optique: Towards OBDA Systems for Industry. In: Proceedings of the Semantic Web: ESWC 2013 Satellite Events, *LNCS*, vol. 7955, pp. 125–140. Springer (2013). DOI 10.1007/978-3-642-41242-4_11
51. Khoussainova, N., Kwon, Y., Balazinska, M., Suciu, D.: SnipSuggest: Context-aware Autocompletion for SQL. Proceedings of the VLDB Endowment **4**(1), 22–33 (2010)
52. Kllapi, H., Sitaridi, E., Tsangaris, M.M., Ioannidis, Y.: Schedule Optimization for Data Processing Flows on the Cloud. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2011), pp. 289–300. ACM (2011). DOI 10.1145/1989323.1989355
53. Kobilarov, G., Dickinson, I.: Humboldt: Exploring Linked Data. In: Proceedings of the Linked Data on the Web Workshop (2008)
54. Kogalovsky, M.R.: Ontology-Based Data Access Systems. Programming and Computer Software **38**(4), 167–182 (2012). DOI 10.1134/S0361768812040032
55. Krivov, S., Williams, R., Villa, F.: GrOWL: A Tool for Visualization and Editing of OWL Ontologies. Web Semantics: Science, Services and Agents on the World Wide Web **5**(2), 54–57 (2007). DOI 10.1016/j.websem.2007.03.005
56. Levie, W.H., Lentz, R.: Effects of text illustrations: A review of research. Educational Technology Research and Development **30**(4), 195–232 (1982). DOI 10.1007/BF02765184
57. Lieberman, H., Paternó, F., Klann, M., Wulf, V.: End-User Development: An Emerging Paradigm. In: H. Lieberman, F. Paternó, V. Wulf (eds.) End-User Development, *Human-Computer Interaction Series*, vol. 9, pp. 1–8. Springer, Netherlands (2006). DOI 10.1007/1-4020-5386-X_1
58. Lopez, V., Unger, C., Cimiano, P., Motta, E.: Evaluating question answering over linked data. Web Semantics: Science, Services and Agents on the World Wide Web **21**, 3–13 (2013). DOI 10.1016/j.websem.2013.05.006
59. Lopez-Veyna, J.I., Sosa-Sosa, V.J., Lopez-Arevalo, I.: KESOSD: Keyword Search over Structured Data. In: Proceedings of the Third International Workshop on Keyword Search on Structured Data (KEYS 2012), pp. 23–31. ACM (2012). DOI 10.1145/2254736.2254743
60. Marchionini, G., White, R.: Find what you need, understand what you find. International Journal of Human-Computer Interaction **23**(3), 205–237 (2007). DOI 10.1080/10447310701702352
61. Martinez-Cruz, C., Blanco, I.J., Amparo Vila, M.: Ontologies versus relational databases: are they so different? A comparison. Artificial Intelligence Review **38**(4), 271–290 (2012). DOI 10.1007/s10462-011-9251-9
62. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language Profiles. W3C Recommendation, W3C (2009). URL `http://www.w3.org/TR/owl-profiles/`
63. Munir, K., Odeh, M., McClatchey, R.: Ontology-driven relational query formulation using the semantic and assertional capabilities of OWL-DL. Knowledge-based Systems **35**, 144–159 (2012). DOI 10.1016/j.knosys.2012.04.020
64. Nunamaker, J.F., Briggs, R.O., de Vreede, G.J.: From Information Technology to Value Creation Technology. In: Information Technology and the Future Enterprise: New Models for Managers, pp. 102–124. Prentice-Hall, New York (2001)
65. Pan, J.Z., Thomas, E.: Approximating OWL-DL Ontologies. In: Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI 2007), pp. 1434–1439 (2007)
66. Popov, I.O., Schraefel, M.C., Hall, W., Shadbolt, N.: Connecting the Dots: A Multi-pivot Approach to Data Exploration. In: Proceedings of the 10th International Semantic Web Conference (ISWC 2011), *LNCS*, vol. 7031, pp. 553–568. Springer-Verlag (2011). DOI 10.1007/978-3-642-25073-6_35
67. van Rijsbergen, C.J.: Information Retrieval, 2 edn. Butterworth-Heinemann (1979)
68. Rodriguez-Muro, M., Calvanese, D.: High Performance Query Answering over DL-Lite Ontologies. In: Proceedings of the Principles of Knowledge Representation and Reasoning (KR 2012), pp. 308–318. AAAI Press (2012)
69. Rodriguez-Muro, M., Calvanese, D.: Quest, a System for Ontology Based Data Access. In: Proceedings of the 9th OWL: Experiences and Directions Workshop (OWLED 2012), *CEUR Workshop Proceedings*, vol. 849. CEUR-WS.org (2012)

70. Rodriguez-Muro, M., Rezk, M., Hardi, J., Slusnys, M., Bagosi, T., Calvanese, D.: Evaluating SPARQL-to-SQL Translation in Ontop. In: Proceedings of the 2nd International Workshop on OWL Reasoner Evaluation (ORE 2013), *CEUR Workshop Proceedings*, vol. 1015, pp. 94–100. CEUR-WS.org (2013)
71. Ruiz, F., Hilera, J.R.: Using Ontologies in Software Engineering and Technology. In: C. Calero, F. Ruiz, M. Piattini (eds.) Ontologies for Software Engineering and Software Technology, pp. 49–102. Springer-Verlag (2006). DOI 10.1007/3-540-34518-3_2
72. Schraefel, M.C., Wilson, M., Russell, A., Smith, D.A.: mSpace: improving information access to multimedia domains with multimodal exploratory search. Communications of the ACM **49**(4), 47–49 (2006). DOI 10.1145/1121949.1121980
73. Segev, A., Sheng, Q.Z.: Bootstrapping Ontologies for Web Services. IEEE Transactions on Services Computing **5**(1), 33–44 (2012). DOI 10.1109/TSC.2010.51
74. Shneiderman, B.: Direct manipulation: A step beyond programming languages. Computer **16**(8), 57–69 (1983). DOI 10.1109/MC.1983.1654471
75. Siau, K.L., Chan, H.C., Wei, K.K.: Effects of query complexity and learning on novice user query performance with conceptual and logical database interfaces. IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans **34**(2), 276–281 (2004). DOI 10.1109/TSMCA.2003.820581
76. Smart, P.R., Russell, A., Braines, D., Kalfoglou, Y., Bao, J., Shadbolt, N.: A Visual Approach to Semantic Query Design Using a Web-Based Graphical Query Designer. In: Proceedings of the 16th International Conference on Knowledge Engineering: Practice and Patterns (EKAW 2008), *LNCS*, vol. 5268, pp. 275–291. Springer (2008). DOI 10.1007/978-3-540-87696-0_25
77. Soylu, A., Giese, M., Jimenez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I.: OptiqueVQS – Towards an Ontology-based Visual Query System for Big Data. In: Proceedings of the International Conference on Management of Emergent Digital EcoSystems (MEDES 2013), pp. 119–126. ACM (2013). DOI 10.1145/2536146.2536149
78. Soylu, A., Giese, M., Jimenez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I.: Towards Exploiting Query History for Adaptive Ontology-based Visual Query Formulation. In: Proceedings of the 8th International Conference on Metadata and Semantic Research (MTSR 2014), *CCIS*, vol. 478, pp. 107–119. Springer (2014). DOI 10.1007/978-3-319-13674-5_11
79. Soylu, A., Modritscher, F., De Causmaecker, P.: Ubiquitous web navigation through harvesting embedded semantic data: A mobile scenario. Integrated Computer-Aided Engineering **19**(1), 93–109 (2012). DOI 10.3233/ICA-2012-0393
80. Soylu, A., Moedritscher, F., Wild, F., De Causmaecker, P., Desmet, P.: 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). DOI 10.1109/ICC.2010.5502398
81. Soylu, A., Skjæveland, M., Giese, M., Horrocks, I., Jimenez-Ruiz, E., Kharlamov, E., Zheleznyakov, D.: 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), *CCIS*, vol. 390, pp. 201–212. Springer (2013). DOI 10.1007/978-3-319-03437-9_21
82. Spanos, D.E., Stavrou, P., Mitrou, N.: Bringing relational databases into the Semantic Web: A survey. Semantic Web **3**(2), 169–209 (2012). DOI 10.3233/SW-2011-0055
83. Spiekermann, S.: User Control in Ubiquitous Computing: Design Alternatives and User Acceptance. Shaker Verlag, Aachen (2008)
84. Staab, S., Studer, R. (eds.): Handbook on Ontologies. International Handbooks on Information Systems. Springer, Berlin, Heidelberg (2009)
85. Studer, R., Benjamins, V.R., Fensel, D.: Knowledge Engineering: Principles and methods. Data & Knowledge Engineering **25**(1-2), 161–197 (1998). DOI 10.1016/S0169-023X(97)00056-6
86. Suh, B., Bederson, B.B.: OZONE: a zoomable interface for navigating ontology information. In: Proceedings of the Working Conference on Advanced Visual Interfaces (AVI 2002), pp. 139–143. ACM (2002). DOI 10.1145/1556262.1556284
87. Ter Hofstede, A.H.M., Proper, H.A., Van Der Weide, T.P.: Query formulation as an information retrieval problem. Computer Journal **39**(4), 255–274 (1996). DOI 10.1093/comjnl/39.4.255
88. Thorpe, S., Fize, D., Marlot, C.: Speed of Processing in the Human Visual System. Nature **381**, 520–522 (1996)
89. Tunkelang, D., Marchionini, G.: Faceted Search. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan and Claypool Publishers (2009)
90. Turk, M., Robertson, G.: Perceptual user interfaces (introduction). Communications of the ACM **43**(3), 32–34 (2000). DOI 10.1145/330534.330535
91. Valencia-Garcia, R., Garcia-Sanchez, F., Castellanos-Nieves, D., Fernandez-Breis, J.: OWLPath: An OWL Ontology-Guided Query Editor. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans **41**(1), 121–136 (2011). DOI 10.1109/TSMCA.2010.2048029
92. Yee, K.P., Swearingen, K., Li, K., Hearst, M.: Faceted metadata for image search and browsing. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2003), pp. 401–408. ACM (2003). DOI 10.1145/642611.642681
93. Zloof, M.M.: Query-by-example: a database language. IBM System Journal **16**(4), 324–343 (1997). DOI 10.1147/sj.164.0324
94. Zviedris, M., Barzdins, G.: ViziQuer: a tool to explore and query SPARQL endpoints. In: Proceedings of the 8th Extended Semantic Web Conference (ESWC 2011), *LNCS*, vol. 6644, pp. 441–445. Springer-Verlag (2011). DOI 10.1007/978-3-642-21064-8_31