SPE 150225

Using Semantic Technology to Auto-generate Reports: Case Study of Daily Drilling Reports

Martin Giese, Univ. of Oslo; Jens I. Ornæs, National Oilwell Varco; Lars Overå, PCA; Inge Svensson, Baker Hughes; Arild Waaler, Univ. of Oslo

Copyright 2012, Society of Petroleum Engineers

This paper was prepared for presentation at the SPE Intelligent Energy International held in Utrecht, The Netherlands, 27–29 March 2012.

This paper was selected for presentation by an SPE program committee following review of information contained in an abstract submitted by the author(s). Contents of the paper have not been reviewed by the Society of Petroleum Engineers and are subject to correction by the author(s). The material does not necessarily reflect any position of the Society of Petroleum Engineers, its officers, or members. Electronic reproduction, distribution, or storage of any part of this paper without the written consent of the Society of Petroleum Engineers is prohibited. Permission to reproduce in print is restricted to an abstract of not more than 300 words; illustrations may not be copied. The abstract must contain conspicuous acknowledgment of SPE copyright.

Abstract

The Daily Drilling Report (DDR), a compulsory report to the Norwegian Petroleum Directorate and the Petroleum Safety Authority (PSA) for operators on the Norwegian Continental Shelf, is an example of a task that typically requires integration of data from a variety of independent sources. Current documentation of DDR (EPIM, 2008) content specifies the reporting format in the form of an XML Schema document (XSD), using textual definitions for the explanation of terms.

We report on a case study that demonstrates the use of semantic technologies for the specification, and possibly also generation, of reports in the Oil and Gas sector based on a case study of Daily Drilling Reports. We discuss the underlying vision, present the case study in some detail and evaluate its results. We furthermore discuss the potential of the method for other kind of reports. Finally we conclude with a few recommendations.

Introduction

The aim of this case study is to investigate the use of cutting edge semantic technologies for the specification, and possibly also generation, of reports in the Oil and Gas sector. This paper concentrates on the vision and main ideas of the approach. In order to give the reader a concrete picture, we have chosen to go to a certain level of technicality. For further technical details, refer to (Overå, 2010).

A report of the type addressed in this study typically contains data aggregated from one or more sources of structured data, and is generated to conform with a predefined format. It is generally accepted that the quality of the reports crucially depends on suitable reporting formats and unambiguous documentation of content. A popular choice is to use an XML-based format, as is done, e.g., for the Daily Drilling Reports (DDR). In the context of DDR, the XML format serves two functions: first, as a transmission format, second, as a documentation format through the associated XML Schema Definition (XSD), along with textual definitions from PCA RDL (PCA, 2008). XML Schema (W3C, 2001) also allows for schema validation for quality control of the data. An overall illustration of the role of the DDR XSD and the PCA RDL in the generation of DDR reports today is given in Figure 1; this is discussed further in the following section.

However, while XML is well suited as a transmission format, it is likely that the level of precision in the documentation can be increased through the use of technologies that implement languages in the so-called "W3C Semantic Web Stack." In addition, these technologies may improve the support for automation of report generation and increase the flexibility for reuse of the report data in other contexts. In particular, the following three W3C-recommended languages are of interest:

- The web ontology language OWL (W3C OWL Working Group, 2009), which can be used to formally articulate definitions and relationships between concepts that are today only semi-formally described in the PCA RDL. Using OWL for this purpose is fully consistent with today's practice of using an XML format for data transmission, but it will replace the documentation function of the DDR XSD and extend the function of the PCA RDL.
- The data model RDF (W3C, 2004) that underlies OWL. RDF is based upon the idea of making statements about resources in the form of subject-predicate-object expressions. In RDF terminology, these expressions are known as triples; a database for storing and retrieving RDF data is called a triple store.
- The query language SPARQL (W3C, 2008), which is designed for querying RDF data.

A word on the relationship between the technologies that support the so-called "W3C Semantic Web Stack" and ISO 15926 (PCA, 2003) is in place. Historically, ISO 15926 was represented in ISO's EXPRESS language. But as W3C technologies came to dominate the field of semantic technologies, ISO 15926 has adapted to the new international trends in technology and now takes full advantage of the progress in W3C-recommended standards and technologies. In particular, there is an OWL version of the core data model ISO 15926-2, the RDL exists in RDF format, and the powerful template constructions described in ISO 15926-7 provide a query language (among other things). There is hence nothing in the present case study that is inconsistent with current and planned developments of ISO 15926, and the results may be smoothly adopted by the ISO 15926 standard.

The case study applied a selection of W3C technologies to a subset of the DDR. We designed a small DDR ontology, mapped the concepts in the ontology up to drilling data provided by an oilfield service company, and used the ontology and mappings to auto-generate a DDR on the basis of the drilling data. Compared to the practice today:

- The DDR ontology and formal queries in the language of the DDR ontology replaces the documentation function of the DDR XSD.
- The mappings from ontology concepts to queries over data sources make the connection between the underlying drilling data and the reported data explicit.

An important virtue of these technologies is that they support declarative methods for report generation. The underlying idea of declarative methods is to specify essential functionalities of a program by explicit descriptions that are external to the program itself, but that the program operates on (so-called executable specifications). A key point is that the specifications shall readily be both understood and modified by users that otherwise don’t need to care about the design of the program as such. The program that operates on the explicit declarations is typically composed of high-level generic software that in principle can be applied in any context. In our case, the declarations consist of three distinct components:

- The DDR ontology (in OWL)
- Queries in the language of the DDR ontology (in SPARQL)
- The mappings from concepts in the DDR ontology to SQL queries over data sources

Generic off-the-shelf components from the semantic technology toolbox, mostly at a prototypical stage, have been used to auto-generate reports from the data.

In the following section, we present what a solution for DDR generation based on declarative methods, in the way sketched above, amounts to and how it contrasts to current practice. In particular, we compare the solutions from four different perspectives:

- Information quality: What are the sources of error?
- Maintenance. How smoothly do the solutions support changes?
- Genericity: Does the report generation practice easily transfer from one type of report to another?
- Reuse: How easy is it to use the reported data in other contexts?

In the “Implementation” section, we review the case study in some detail, focusing on technology and methods, and present our contributions in with a certain degree of technical detail. Readers who want to be spared for technicalities may safely skip this section and move on to the next.

In the “Evaluation” section, we put the results of the case study in perspective. First, we discuss the limitations of the implementation and challenges that we had to face. Second, we discuss the potential of the method for other kind of reports, like production data.

We conclude with a section giving a few recommendations for the use of this kind of technology for reporting purposes.

Generating Daily Drilling Reports

The current practice

Current practice for DDR generation is illustrated in Figure 1. Data is aggregated from one or more data sources; typically the sources are relational databases accessed using SQL queries. The XML document that constitutes a DDR is generated from a program, typically written in Java. The program interacts with a software module that generates syntactically correct XML documents, through a predefined API. The developers are guided by two kinds of documentation in the design of the program: first, the XSD itself; second, the definitions in PCA’s RDL that the documentation links up to. The PCA RDL contains definitions in natural language, as well as some class relationships, e.g., to subclasses and superclasses.

For generation of DDRs, this is a robust setup. Besides the quality of the data in the databases, the quality of the reported data depends first and foremost on the programmer’s ability to recast the documentation of the concepts in the DDR XSD into correct queries over the data sources. With this information at hand, writing correct program code is straightforward.

However, generating reports through a manually written program puts some burden on maintenance. If data sources change, or if the report format changes, one has to modify the program accordingly, and even for small programs like those that generate DDRs, this process can be expensive and possibly also error-prone, in particular if one has to modify other developers’ code. Moreover, the program is not easily applicable to other contexts; a program that generates another kind of report than DDR will most likely have to be developed from scratch. It is for these aspects that the use of declarative methods is promising.

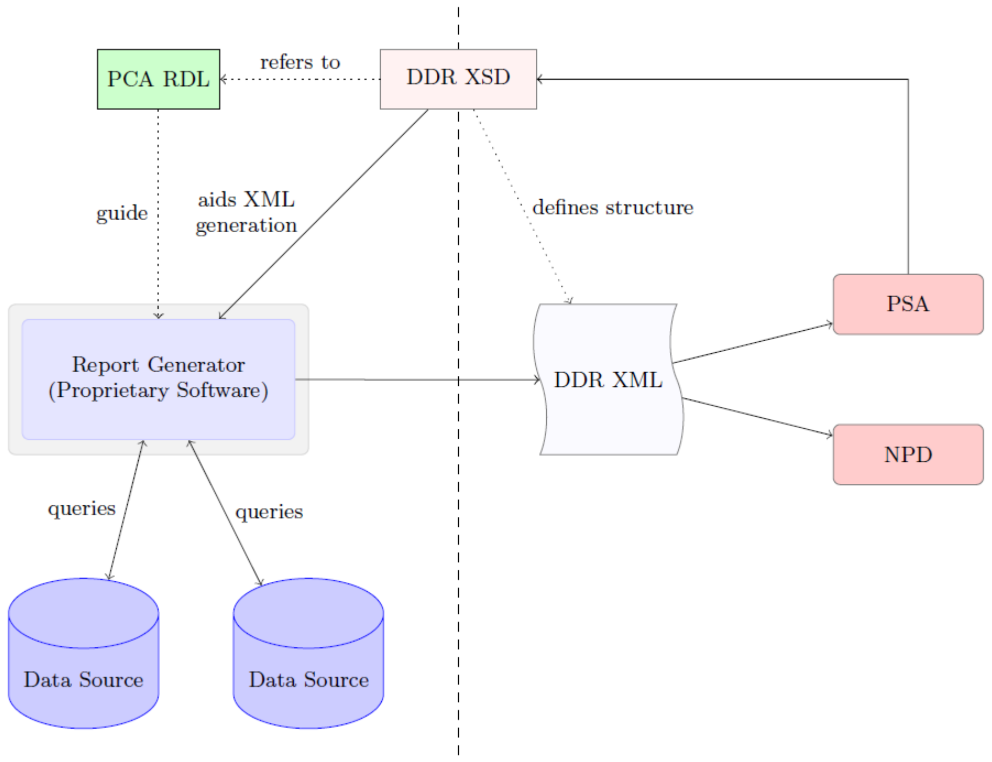

Figure 1: Current practice for DDR generation

The potential for reusing reported data in other contexts, or in combination with other data, depends on the quality of the data descriptions. The DDR data are currently described by the DDR XSD. If DDR data are to be combined or integrated with other data, one will often have to relate the data through an analysis of the relationship between data descriptions, and for this purpose the information in the DDR XSD leaves out too much information. I.e., information that is important for integration is not made explicit, but assumed to be known from the context of use, and this makes integration difficult.

A solution that exploits W3C technology

An architecture for a solution that exploits both W3C semantic web technology and the ISO 15926 toolbox is depicted in Figure 2. Compared to the current situation there are four main differences.

First, the role that PCA RDL serves in Figure 1 is replaced by a further formalization of the definitions and descriptions in the RDL in the form of:

- An ISO 15926 DDR ontology
- DDR templates
- The combination of the extension of ISO 15926 and the definition of templates amounts to an ISO 15926 representation in which all the concepts and relationships that describe the content of the DDR, have been fully formalized in a machine processable way. In ISO 15926, there is a division of labor between core ontology concepts and templates, but where one draws the borderline is not essential for this setup as both the ontology and templates are parts of the construction.

Second, while the DDR XSD in Figure 1 contains just a hyperlink to PCA RDL, the proposed architecture in Figure 2 makes the link to PCA RDL explicit and precise. This is achieved by specifying the content of each data entry in the DDR in the form of formal SPARQL queries formulated in the vocabulary of the ISO 15926 DDR ontology and templates. The queries are written in a predefined and machine processable format.

Third, mappings from queries over data sources to concepts in the ISO 15926 DDR ontology are made explicit in a machine processable format. This opens in particular the possibility of linking up to several data sources. Assuming that data sources have an SQL query interface the mappings will map concepts in the ontology to SQL queries over the data sources, or more generally, map queries formulated in the ontology language to SQL queries.

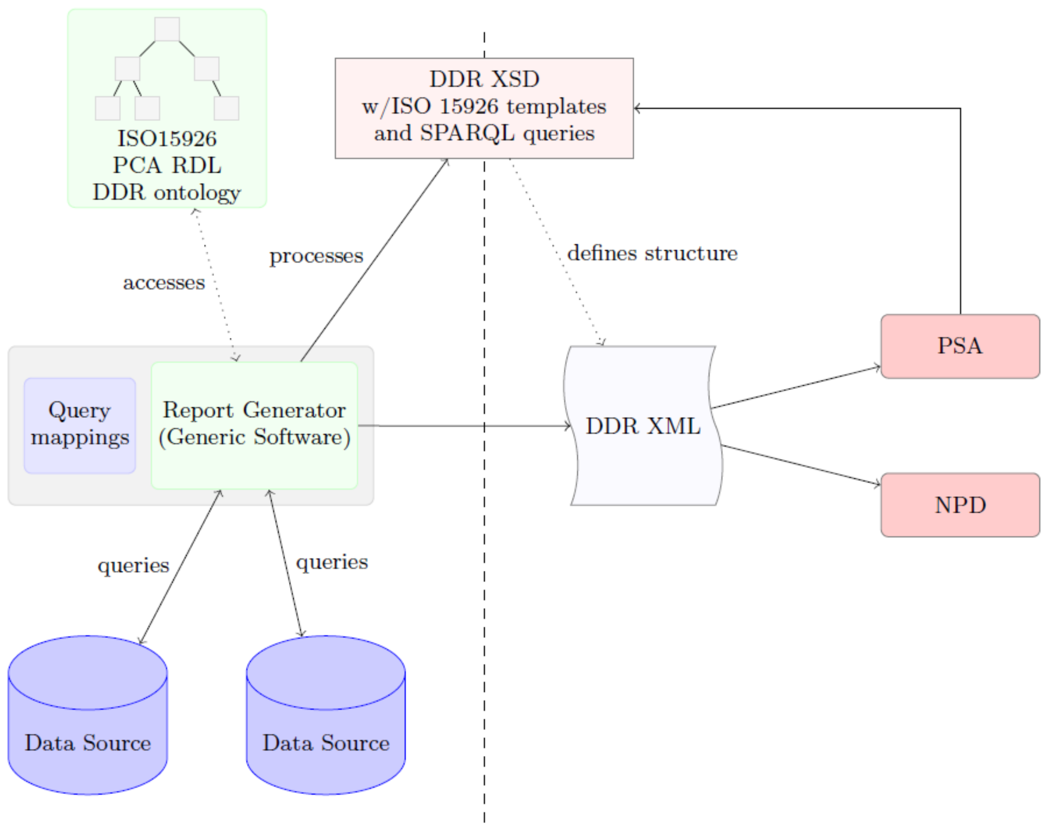

Figure 2: Architecture for a solution that exploits W3C technology

Fourth, in Figure 2 the software is composed of generic components. The XSD with queries can straightforwardly be processed by a program that parses the XSD and the queries, executes the queries, and outputs the results in the form of an XML document that complies with the XSD. The task of generating queries to the underlying data sources from the queries in the XSD can be handled by a so-called Query Rewriter. Technology for this has been produced recently from cutting edge research, but commercial products are likely to become available as soon as there is demand for them. In Figure 2, the Query Rewriter takes as input queries in the ISO 15926 vocabulary, the ISO 15926 DDR ontology, and the mappings.

An important insight from research in Description Logics, the logical formalism that underlies OWL, is that the choice of language in which the ontology (i.e. the "ISO 15926 DDR ontology") and the associated queries (i.e. the SPARQL queries in the "XSD w/queries") are represented matters. Broadly speaking, there is a tradeoff between the expressivity of the ontology language and the computational cost of the key inference steps associated with constructs in the ontology language. The Description Logic DL-Lite (Artale et al, 2009) has been specifically designed for use in query rewriting applications, and now underlies the new OWL 2 profile OWL-QL (W3C, 2009). OWL-QL is the ontology language that W3C recommends for data-intensive applications.

While the theory behind DL-Lite and query rewriting is now fairly well understood, query rewriting tools for DL-Lite are still at a prototype stage. In our case study, we applied the rewriting tool QuOnto (Acciarri et al., 2005), that was developed in a recent EU-funded research project.

In the architecture in Figure 2, there are three explicit specifications that determine the output: the ontology, the queries in the XSD, and the mappings. Here are the main consequences of the architecture compared to today's practice:

- Information quality. The architecture has reduced the need for manually interpreting the XSD and designing a program that generates the reports. The mappings and queries are specified outside the program code. Moreover, the generic components are as a rule very reliable software. Hence one potential source of error is eliminated. For the DDR, this is in itself not so much of an improvement, but this aspect may be more significant in situations where the domain is less well-structured than is the case for drilling.
- Maintenance. Semantic technology comes with the promise of simplifying maintenance. This is due to the role that explicit specifications and declarative methods play in the architecture. In principle, if there is need for a change, one just has to modify one of the specifications and change that specification locally: If the data sources change, the mappings must be updated. If the report format changes, the XSD with its associated queries must change. If new concepts need to be accounted for, the ontology must be changed to accommodate for this. Moreover, the specifications are more easily accessible than program code and can both be read and understood by more people, and in particular by domain experts. This may facilitate the challenge of maintenance significantly.

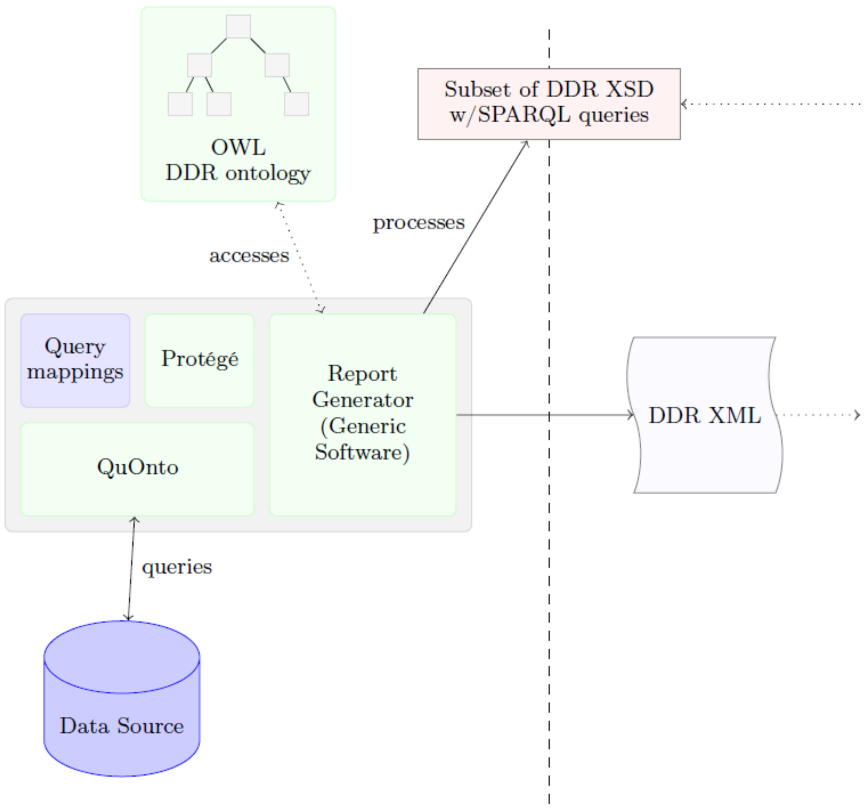

Figure 3: Implemented solution

- Genericity. The generation of one type of report can in principle easily transfer to another, as it suffices to change the mappings, queries, and ontologies. In principle, this architecture for DDR generation can be used in another context without further modifications.
- Reuse. The quality and size of the ontology determines how easy it is to use the reported data in other contexts, since the ontology is the main vehicle for alignment. A great advantage of the architecture in Figure 2 is that the ontology component is explicit and plays a direct role in the execution of the program. This increases the degree to which the data and specifications can be machine manipulable, which is significant for data reuse.

Implementation

In this section we describe the implementation of the use case in some detail. The implemented solution is visualized in Figure 3. Note that the "Report generator" of Figure 2 has been implemented using the tools QuOnto and Protégé (Stanford, 1995), in addition to a simple Java program that was made for this use case, but that is in fact completely generic. Compared to the general architecture in Figure 2, two key simplifications have been made.

- The relationship to ISO 15926 is shallow. Due to limited resources for ISO 15926 implementation, just a very simple OWL QL ontology was designed.
- There is just one data source. It is in principle straightforward to extend the use case with multiple data sources, but this requires a data federation tool like IBM's Clio, which was not available to us.
- We implemented a reresentative subset of the DDR XSD. To the best of our knowledge the complexity of the subset of DDR XSD that we implemented is representative for the whole.

The implementation succeeded in producing a correct DDR XML report from the available data. Below we present details of the components of the implemented solution.

DDR XSD/XML

XML Schema (XSD) is a W3C standard which defines what an XML document should contain and how the contents should be structured. The Daily Drilling Report is defined by an XSD file published by EPIM (Exploration & Production Information Management Association). This is what operators today utilize in the creation of their DDR documents. In this case study, we have added SPARQL queries to the XSD that are used to query in the vocabulary of the OWL DDR ontology, as explained in the following sections.

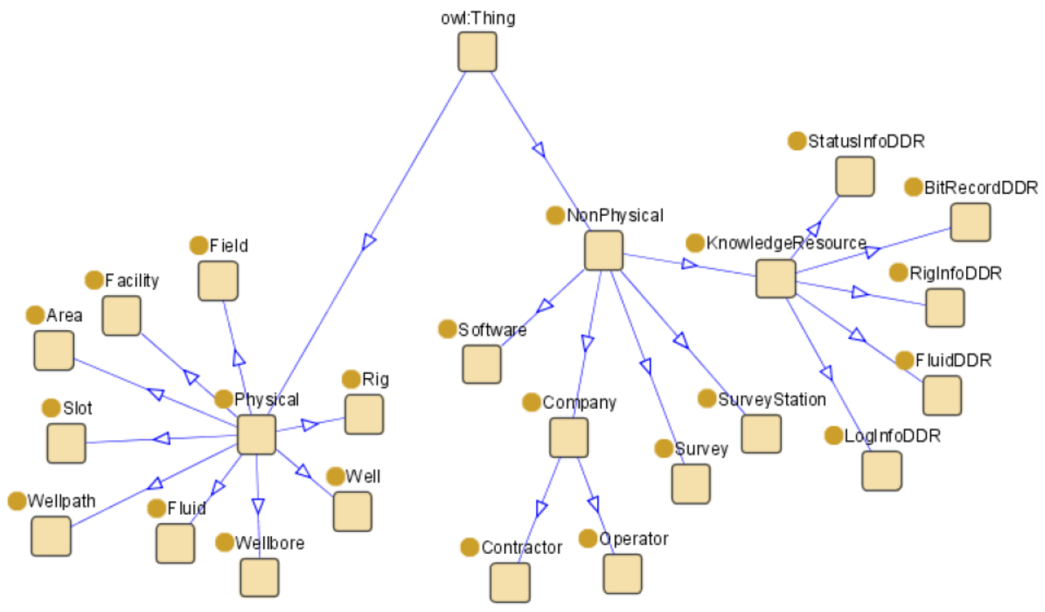

Figure 4: The OWL DDR ontology

XML is a much-used format for storing and transferring data in a human-readable format. For DDR, the structure of the XML document is defined by the DDR XSD. Also in this use case we use the XSD file to determine the structure of the XML file being created; hence the resulting DDR XML document is no different from a document created using other software.

OWL DDR ontology

The OWL DDR ontology for the use case is depicted in Figure 4. It was constructed by extracting relevant tags from the DDR XSD and adding some extra structure.

As explained previously, the OWL DDR ontology is a formal model of the concepts used in a Daily Drilling Report. The main purpose of the ontology is to provide a vocabulary for the SPARQL queries declared in the DDR XSD file. This configuration is part of what makes the method fully declarative, in that no actual software has to be altered in order to facilitate changes to the report format (in this case DDR). The relationship between the OWL ontology and SPARQL can be compared to an SQL database schema and SQL queries. Another important application of the ontology is to facilitate reuse of data in integration. For this, a strong connection to a larger ontology such as ISO 15926 is necessary. While this has not been performed, it is a natural extension of our use case and is crucial in future aligning of data from multiple sources gathered for different purposes.

An OWL ontology aims at providing an unambiguous description of a particular domain of interest. This domain of interest can be small and consist of just a few classes, or large and consist of thousands of classes. The DDR ontology was created for this use case as an unambiguous nexus between the DDR and local data sources, enabling various local data sources to be interpreted so that the meaning is unambiguous when inserted into a DDR document. The ontology is created reflecting the data entries in the DDR XML; e.g., the Wellbore class in Figure 4 refers to a wellbore in DDR. The SPARQL query that asks for data about wellbores should thus be straight-forward to understand. The following sections contains more information on these SPARQL queries.

The OWL DDR ontology is a formal description of the concepts needed to create DDR documents. It serves both as a vocabulary for SPARQL queries and potentially as platform for integrating data through ISO 15926.

SPARQL-annotation of the DDR XSD

As one of the three components required for making a declarative report generator, the SPARQL queries have been expressed in a way that makes them easily accessible by the generic report generating software. Embedded in the XSD file, the SPARQL queries are extracted by the software and then processed by QuOnto conforming to the DDR ontology. This is an example of an XSD with embedded SPARQL queries:

```
    <xsd:complexType name="cs_drillReportSurveyStation">
      <xsd:annotation>
        <xsd:documentation>WITSML - Trajectory Station
          Component Schema</xsd:documentation>
        <xsd:appinfo>Modified-in-version=1.4.0,
          By-issue=1.3.1-33, Change=Added</xsd:appinfo>
        <xsd:appinfo>
          <sparql_query:quontoquery>
            SELECT ?md ?tvd ?azimuth WHERE {
              ?x :md ?md .
              ?x :tvd ?tvd .
              ?x :azimuth ?azimuth . }
          </sparql_query:quontoquery>
        </xsd:appinfo>
      </xsd:annotation>
      <xsd:sequence>
      ....
```

Everything besides the `xsd:appinfo` element and its contents (printed in boldface) is identical to the standard DDR XSD. The lines in blue are a SPARQL query that extracts data included in the DDR-defined complex data type "`cs_drillReportSurveyStation`"; in other words DDR data about survey stations. Without going into too much details about SPARQL, observe the following:

- `SELECT` is a SPARQL keyword used to select the columns of the answer table, similar to SQL.
- `WHERE` is a SPARQL keyword that precedes a list of "triple patterns," in which
  - Each line is a *triple*, or *statement*, made up of a subject, a predicate and an object, terminated by a `.`
  - Words starting with `?` are variables. These are the slots that the query engine will substitute with data values that match the statements.
  - Words starting with `:` are terms that exist in the OWL vocabulary created in step 1
  - Each answer to the query consists of an assignment of values to the variables, that make all triples of the query match the data base. E.g., the triple `?x :md ?md` alone will list all values `?x` and `?md` which are connected by an `:md` predicate in the data base.

In the use case SPARQL queries are declaratively expressed and added to the DDR XML Schema. The Report Generator generic software extracts these queries and passes them on to QuOnto.

SQL server

SQL is a common query language for relational databases. SQL servers are widely used commercially for storing data. It is thus natural to use an SQL server as the storage of the data used to create the DDR documents.

In this use case, we use a MySQL database to store raw data and SQL queries to fetch these data. However, all SQL queries are specified explicitly within mappings. The SQL queries sent to the relational database are created automatically from the mappings by the query rewriter in QuOnto, which converts SPARQL queries to SQL queries.

For our case study, we populated the MySQL database with data provided by an oilfield service company. The limited set of data we were provided with dealt primarily with well paths and measurements relevant to these.

SQL queries that extract data are not hard-coded and executed from proprietary software, but are specified in explicit mappings and then created automatically from SPARQL queries by QuOnto.

QuOnto

This is a reasoner and query rewriter for the ontology language DL-Lite. DL-Lite refers to the Description Logic which underlies the OWL profile OWL-QL, and is as such a subset of OWL. This subset has the important property that it is very efficient when querying large amounts of data. In this use case, the reasoning capability of QuOnto is not used to any great extent, and the only reasoning conducted is a consistency check.

The main task of QuOnto is to handle the SPARQL queries against the ontology and translate them through mappings into SQL queries.

QuOnto is the OWL reasoner that in this use case converts SPARQL queries to SQL queries.

OBDA stands for Ontology Based Data Access and is a term accompanying many of the components of QuOnto. The OBDA server is where the QuOnto reasoner is executing. It can be seen as a wrapper and interface for the QuOnto reasoner that processes incoming requests and passes them on to QuOnto. Also, the OBDA mappings from SPARQL to SQL are passed through this server to QuOnto.

The OBDA server hosts the QuOnto reasoner.

OBDA mappings
OBDA mappings are the third component required for the declarative document generation. These mappings map SPARQL queries to SQL queries (and potentially other kinds of queries). They serve a crucial function in the use case, as much relies upon the ability to properly translate a SPARQL query to (a set of) SQL queries. Once that job is done, it is a simple task for any kind of SQL server to fetch the queried data and return them.

Two typical mappings used for this use case look like this:

```
<mapping id="Wellbore(Class)">
  <CQ string="Wellbore(getWellbore($ID))"/>
  <SQLQuery string="SELECT ID FROM wellbore"/>
</mapping>

<mapping id="tvd(Property)">
  <CQ string="tvd(getSurveyStation($Wellbore,$MD),$tvd)"/>
  <SQLQuery string="SELECT DISTINCT(Wellbore) AS Wellbore,
                    MD, tvd FROM trajectoryStation"/>
</mapping>
```

Here we see two mapping entries, the first for an ontology class and the second for an ontology property.

- The `id` can be anything as long as it is unique. A meaningful name should though be chosen.
- The `CQ` is the "Conjunctive Query" part which refers to a SPARQL query. Note that the first mapping above has a unary predicate which in OWL means a class, while the second mapping has a binary predicate which means OWL property.
- The `SQLQuery` is the SQL query necessary to extract the information needed to satisfy the Conjunctive Query.

OBDA mappings are used to define translations from SPARQL queries to SQL Queries.

Protégé
Protégé is a popular editor used for creating ontologies. The DDR OWL ontology as well as the ontology visualization shown in Figure 4 was created using this tool.

Besides being an ontology editor, Protégé is used for another purpose in this use case: The query rewriting software QuOnto is implemented as a plug-in to Protégé. It obtains the ontology and mappings via the Protégé framework. This is a technicality specific to QuOnto, and will be different for future query rewriting tools.

Protégé is used as an ontology editor and, through the use of a plug-in, as a link between the report generating software, the ontology and the QuOnto server.

Report Generating Generic Software
This is the piece of generic software that connects the other components and that is executed to generate the finished DDR XML document. The tasks of the software in this particular implementation ordered chronologically in order of application are:

- Extracting the queries from the DDR XSD extended with SPARQL queries
- Sending the SPARQL queries to Protégé (from where they are passed to QuOnto, and on to the database)
- Receiving the results from Protégé
- Using these results to create the DDR XML document

The software to conduct these tasks has mostly been created from scratch in Java. The part that interfaces with QuOnto uses software from the QuOnto creators modified to suit this use case.

The Report Generator Generic Software is the executing component that initializes the whole report generation. It extracts queries from the XSD, passes them to Protégé, fetches the results and uses these to create the DDR XML document.

Summing up

Here we provide a short summary of the use case components in the form of a step-by-step overview of the report generation based on our method. Note that the order of the tasks need not be exactly like this.

1. We first created the DDR OWL ontology based on the contents of the DDR XSD, while keeping in mind how the ontology should be queried using SPARQL.
2. Then we expanded (annotated) the DDR XSD with SPARQL queries
3. We set up a (MySQL) SQL database and populated it with data provided by an oilfield service company.
4. We created the mappings from SPARQL to SQL
5. We set up a QuOnto OBDA server for translating queries using the mappings
6. We set up the required Protégé plugin to communicate with the QuOnto OBDA server and attached the mappings

Finally, we created generic report generator software which parses the DDR XSD file and retrieves the SPARQL queries needed. It passes these queries through Protégé to the QuOnto OBDA server which produces SQL queries which are then used to query the SQL server. The resulting answers are used to create the DDR XML file.

Evaluation

Limitations of the use case

Only a fragment of DDR

The available resources made it necessary to focus on only a fragment of DDR. A significant part of the work went into developing the method and simply getting the tools to function properly together. We did not have sufficient resources available to acquire more data and extend the use case.

That being said, now that the methodology is in place and the tool chain for this implementation is established, expanding the use case to encompass all of DDR is in principle straightforward. The expansion mainly concerns the three components mentioned earlier, that are required for the declarative report generation: ontology, SPARQL queries and query mappings.

DDR OWL ontology

The ontology we created performs as it should. It contains the concepts and relations necessary to facilitate SPARQL queries, and provides the formal model for QuOnto.

The DDR ontology we created does not have a strong connection to ISO 15926, in contrast to the ideal solution shown in Figure 2. It does, however, contain concepts that are easily aligned with ISO 15926, and a full alignment is clearly a priority for a full scale implementation of the use case. It will not only give additional formal context to the data used to create Daily Drilling Reports, but it will also enable integration and exchange with other data.

Although the ontology was represented in OWL QL, we did in fact make little use of the strengths of this fragment of OWL. In particular, we did not require complex SPARQL queries, and therefore had no need for the advanced query rewriting capabilities of QuOnto. This is, however, due to our choice of using DDR as our use case. DDR concepts are highly specialized and do not form a complex structure of concepts; therefore there is no real need for reasoning capabilities that allow us to, e.g., shift between different levels of abstraction. There is however every reason to believe that advanced query rewriting works as promised, and that it has application in more complex reporting scenarios.

Difficulties with the tools

As some of the tools we used are prototypes stemming from academic research and not of commercial quality and reliability, setting them up and getting them to work properly was demanding. QuOnto in particular was difficult to get to function as required. A lot of time was spent on configuring it, creating the query mappings, and connecting it all through the Protégé OBDA plug-in. Part of the problem was that the interface connecting the various components was incomplete and did not support all of the constructs of OWL. This meant that a lot of tweaking had to be done to get everything working properly. This is essentially unavoidable in prototypical tools, but it should be easy to improve on in future tools of commercial quality.

Report Generator Generic Software

In the ideal solution shown in Figure 2, the box with generic software is meant to handle all of the interfaces towards the other components which are

- The DDR XSD (XML Schema) with queries
- The DDR OWL ontology

- The query mappings
- The Data sources (often SQL)
- The DDR XML document (created by the software)

This was however not possible to accomplish in the implemented solution, due to limitations inherent in working with QuOnto. What needed to be done was instead to split the generic software component into several parts which handled separate tasks. These were

- QuOnto, which is the reasoner and query rewriter. It receives SPARQL queries and rewrites them to SQL and then queries the SQL server.
- Protégé, which handled loading the query mappings and connecting to the ODBA server.
- Generic software. Instead of linking to the five components listed above, in our implementation the generic software only connected directly to the DDR XSD, Protégé and the DDR XML.

Although the implementation had more components than the ideal solution shown in Figure 2, it is likely that the OWL reasoner which in this case was QuOnto will not be a part of the generic software, but rather an external component which the generic software interfaces against. In that regard, the thing that complicated this implementation was the need to include Protégé. Interfaces against the five components listed above will in any case be necessary, and as long as they facilitate the declarative report generation, splitting the generic software component into several parts should not be a problem.

Potential for other kind of reports

The methodology that was applied in the use case has its main strength in one of the following situations:

- The reporting format changes frequently. Then one can update the queries.
- The data sources change frequently. Then one can update the mappings.
- The description of the reported data is complex. Then there is a real need for an ontology.

In all these situations one can profit from the explicit representation of an ontology, queries and mappings. On this background, DDR is not an ideal target for the method although, thanks to the available XSD, it was the natural candidate for a "proof of concept" implementation.

An application domain for which the method seems particularly promising is environmental reports, since one can expect the available models, data sources, and reporting formats in this area to be in flux at least for the next few years.

Recommendations

We have presented a case study of an auto-generation of a Daily Drilling Report using declarative methods and explicit representation of an ontology, queries and mappings. In evaluating our case study, we should separate three distinct aspects:

- The declarative specification of the content of the DDR through an ontology and queries in the XSD
- The declarative specification of the relationship between ontology concepts and the structure of the underlying data source through the mapping
- The use of query rewriting software to fully automate the report generation

All three aspects contribute to data quality, but they also have a cost associated with implementation.

Our case study shows that a full auto-generation of reports is within reach, although the software is not yet sufficiently mature. But first and foremost it shows how one can separate ontologies, queries and mappings, and give information that today is hidden in program code an explicit representation. This is of value also if the reports are not auto-generated.

Based on the lessons of the use case, we conclude with a few recommendations:

- First, extend ISO 15926 with all the concepts used in the specification of the content of a report in an explicit way. For the DDR, the small ontology designed in this use case can serve as a starting point.
- Second, design templates for the specification of the contents in the reports. This opens for a validation of the types of data that go into the report against the PCA RDL, which in itself can give a significant increase in data quality.
- Third, keep the XSD format for report specification, but add templates to the XSD entries to increase the level of precision of the specification. The advantage of keeping the XSD format is that this is widely supported by current software.

The possibility for auto-generation requires sophisticated use of semantic technologies. In particular the mappings must be of a specific form, the ontology must be in OWL-QL format, the queries must be written in SPARQL. Moreover, the tool support is yet not mature enough for commercial use. So although this is an interesting and promising possibility, we recommend a "wait and see" attitude.

An important step toward a fully declarative solution is the mappings from vendor systems to concepts in the ontology. If the main software vendors can be pushed to make these mappings explicit, this may in itself have significant impact on data quality. A standardized format for query mapping representation will be helpful to this end.

References

Acciarri et al., A. (2005). QUONTO: QUerying ONTOlogies. *In Proc. of AAAI 2005* , 1670-1671.
Artale et al, A. (2009). The DL-lite family and relations. *J. Artif. Int. Res. 36.1* (pp. 1-69). issn: 1076-9757.
Calvanese et al, D. (2008). Data Integration through DL-LiteA Ontologies. In K.-D. Schewe, & B. Bernhard Thalheim (Ed.), *Revised Selected Papers of the 3rd Int. Workshop on Semantics in Data and Knowledge Bases (SDKB 2008).* (pp. 26-47). Vol. 4925. Lecture Notes in Computer Science. Springer.
EPIM. (2008). *EPIM Reporting Forum - Drilling*. Retrieved December 2011, from EPIM: http://drilling.posccaesar.org/
Overå, L. (2010). Semantic technology in the oil & gas drilling domain. *MSc thesis, Dept. of informatics, Univ of Oslo* .
PCA. (2003). *ISO 15926*. Hentet December 2011 fra PCA: https://www.posccaesar.org/wiki/ISO15926
PCA. (2008). *PCA Reference Data Library*. Retrieved December 2011, from POSC Caesar: https://www.posccaesar.org/wiki/Rds
Stanford. (1995). *Protégé*. Retrieved December 2011, from Stanford: http://protege.stanford.edu/
W3C. (2009). *OWL 2 Web Ontology Language Profiles*. (B. Motik et al., Editor, & D. Calvanese et al, Producer) Retrieved December 2011, from W3C: http://www.w3.org/TR/owl2-profiles
W3C OWL Working Group. (2009). *OWL 2 Web Ontology Language Document Overview*. Retrieved November 2011, from W3C: http://www.w3.org/TR/owl2-overview/
W3C. (2004). *Resource Description Framework (RDF).* (R. W. Group, Editor) Retrieved December 2011, from W3C: http://www.w3.org/RDF/
W3C. (2001). *Schema*. Retrieved December 2011, from W3C: http://www.w3.org/standards/xml/schema
W3C. (2008). *SPARQL Query Language for RDF*. (E. Prud'hommeaux W3C, & A. Seaborne Hewlett-Packard Laboratories, Editors) Retrieved December 2011, from W3C: http://www.w3.org/TR/rdf-sparql-query/