SPE 112267

Semantic web technologies for smart oil field applications

Ramakrishna Soma[1], Amol Bakshi[1], Viktor Prasanna[1], Will Da Sie[2], Birlie Bourgeois[2]

[1]University of Southern California, Los Angeles CA, USA
[2]Chevron Corporation, USA

This paper was prepared for presentation at the 2008 SPE Intelligent Energy Conference and Exhibition held in Amsterdam, The Netherlands, 25–27 February 2008.

Abstract

In model based oil field operations, engineers rely on simulations (and hence simulation models) to make important operational decisions on a daily basis. Three problems that are commonly encountered in such operations are: on-demand access to information, integrated view of information, and knowledge management. The first two problems of on-demand access and information integration arise because a number of different kinds of simulation models are created and used. Since these models are created by different processes and people, the same information could be represented differently across models. A unified view of the models and their simulations is desirable for decision making, and thus the necessity for information integration. Knowledge management refers to a systematic way to capture the rationale (knowledge) behind the various analyses performed by an engineer and decisions taken based on the analyses. It is critical to capture this knowledge for auditing, archiving, and training purposes. In this paper, we propose the application of semantic web technologies to address these problems. The key elements of the semantic web approach are the ontologies or the information schemas that model various elements from the domain, and a knowledge base (KB) which is a central repository of the instance information in the system. We present a modular approach for organizing the ontologies and outline the process that was followed to define the ontologies. We also describe the workflow that was used to populate the KB and briefly discuss some of our prototype applications that address the problems mentioned above. Based on our experience, semantic web technologies appear to be a highly promising approach to deal with these information management issues in the oilfield domain, although performance and tool support remain the key areas of concern at this stage.

1. Introduction

The work described in this paper is part of the Integrated Asset Management (IAM) project at the Chevron-funded Center for Interactive Smart Oilfield Technologies at the University of Southern California, Los Angeles[17] . The current focus of the IAM project is on enabling model-driven reservoir management. As a motivating example, consider a typical oil-field operation setting for a green field. Since little or no performance related data for the field exists, the production engineer has to rely on simulations for making the initial set of asset development decisions. Different simulation models of the oil-field are created and used – these include earth models, reservoir simulation models, network models, integrated (coupled) simulation models, etc. These simulation models are built and used at different times, different locations, and by different asset team members – earth scientists, reservoir engineers, production engineers, asset managers, etc. A particular member of the team (say, the reservoir engineer) is typically an expert in a particular modeling and simulation technology and intimately familiar with certain software toolkits in that domain. This also means that models, workflows, and results created by other software tools in other domains are not usable and accessible by that expert. As a result, the insights and understanding of a team member in one role (say, geologist) are not fully utilized by another role (say, production engineer). Moreover, these simulation models could be constantly modified as new data is continuously produced in the oil-field and interpreted by one or more members of the asset team. In this situation, changes made to the model(s) by one team member should be immediately communicated to other team members who may be using that model as the basis of scenario planning and forecasting, or who may need to modify their own models to match the updates. Three of the many problems that are observed in this setting are:

- Efficient access to information: No engineer has complete knowledge of all the data in the system and finding the relevant piece of information required to make a decision is a challenge.

- Unified view of the information: Every simulation model, models one facet (reservoir, network etc) of the oilfield in detail. However, a unified view of the information related to the asset elements is generally not accessible from one place or application.
- Knowledge management: As the models are constantly being calibrated and decisions are taken, the rationale (knowledge) behind the changes and decisions are generally lost. Such knowledge could be extremely useful for auditing the decisions made and also to train new engineers.

Similar problems are observed widely in IT enabled businesses and IT enabled science (e-science), as large scale instrumentation of physical and non-physical elements, have led to increasing amounts of data being generated and used. Users are increasingly overwhelmed by the large volumes of data generated and systems that help them quickly search for the right data and access it are the need of the hour. Such systems, *knowledge bases*, should capture the key information in the data objects, the business context in which the data was created, the context in which the information in the data set can be applicable, etc. This makes it possible for the user to search for information using terms relevant to and within the context of the business.

The problem of finding information in a system with large amounts of largely unstructured data is also seen on the internet. To address this problem the W3C organization, the leading standards definition body for the web, has proposed a set of standards for data modeling, knowledge capture, and semantic querying and retrieval of data- commonly called the *semantic web* standards. In this paper we examine the applicability of these standards for the problems described above in real time reservoir management. The main motivation for employing these technologies for our problem setting is that they provide a simple and intuitive data model, are designed for collaborative and distributed evolution, and provide capability for inferencing which allows conformant tools to infer additional facts from the data provided by the user.

Similar work in applying semantic web technologies to address the information integration and the knowledge management problems in the oil and gas industry have been proposed [14] [15] [16] . An industry wide ontology based on semantic web technologies, called the *Oil and Gas Ontology (OGO)*, has been proposed by POSC Caesar, in an attempt to provide standard means for data integration within and across business domains [14] . In [15] the author proposes different use-cases for the application of semantic web technologies in the oil and gas industry. A system for knowledge management for drilling experiences is presented in [16] In contrast, we apply this technology to address the problems in the reservoir management and real-time oil field operations setting. In particular, we present the issues in developing an ontology, acquiring knowledge for the knowledge base and some examples of applications that help solve the integration and knowledge management problems that we have encountered.

The rest of the paper is organized as follows. In Section 2, we introduce the background technologies and terminology used in the rest of the document. In Section 3 we highlight the key application areas for ontologies and semantic web technologies in this problem setting. Sections 4 and 5 outline the key elements of the ontologies and the design methodology that we have used to develop our solution. In Section 6 we describe some workflows and tools from our prototype that uses the ontologies. We finally present some discussions, lessons learnt and future directions for our work.

2. Background

Ontologies and Knowledge Bases

An ontology is a shared representation or a *data model* of a set of concepts in a domain and the relationships between them [2] . An ontology has been commonly used to solve two important and related problems occurring in large organizations: information integration and knowledge management. The information integration problem occurs as, different systems and databases represent and store information in different ways. These differences are not just *syntactic*, i.e., using different technologies (XML, RDBMS Object Oriented etc.) to represent and store the information, but also *semantic* in nature. Semantic differences in information representation means that data is named or encoded differently in different data sources. E.g., a very commonly observed phenomenon in oil-field operation is that the same entity (say a well), is called by different names in different sources (aliasing). As another example, different unit systems are used to represent information in different information systems. Such issues have been successfully addressed by using an ontology. The definitions of the concepts and relationships are only a means to categorize and capture the *real instance* data in the domain. A data store which contains data that are instances of the concepts in the ontology is called a *knowledge base*. The ontology can be considered to be the *schema* for the knowledge base. Shared data models like PRODML and WITSML, defined by Energistics (formerly POSC), also address similar problems and can also be considered as ontologies. In this paper, we consider ontologies defined and encoded using semantic web technologies and argue that some of these applications are better implemented using such data models.

Ontologies are captured or represented by ontology languages. Since an ontology is used as a shared representation, it is desirable that it be represented in a language which is non-proprietary or *open*. Further, the ontology language must provide enough features to represent rich definition of concepts. Thus, another important requirement for an ontology representation language is that of *expressiveness*. Finally, the ontology language should support the ability to represent and query instance data.

Semantic Web Technologies

Tim Berners Lee, the pioneer of the World Wide Web, has put forth the idea of semantic web, as the next generation web where computers *become capable of analyzing all data on the web [1]* . The World Wide Consortium (W3C), which is the main body defining the standards for the web, has proposed a set of standards that build upon the currently prevalent ones like XML and address some of the key needs of such a semantic web. These standards address such areas as languages for rich knowledge representation, querying, security, etc. Figure 1 below shows the semantic web standards being defined by the W3C consortium. In this paper we will focus on the standards highlighted in the figure to define ontologies and create knowledge bases.

RDF

Resource Description Framework (RDF) is a World Wide Web Consortium (W3C) specification originally designed as a metadata model but has been used as a general method of modeling information. The RDF paradigm is based upon the idea of making *statements* about *resources*. Thus, the basic unit of data representation in RDF is a *statement* or *triple*, which is of the form *subject-predicate-object*. A set of related statements form an RDF graph. Although the graph based RDF paradigm can be encoded in different ways, the most common and important way of encoding it is as a XML document, using a well defined convention.

Two important specifications closely related to RDF are the RDF Schema (RDFS) and SPARQL. RDFS is used to define the meaning of the concepts used in an RDF document. The relationship between RDFS and RDF is similar to the relationship between XML schema and XML. The difference between XMLS and the RDFS is that unlike XMLS which only allow the definition of syntactic structures, RDFS supports the notions of class, class hierarchies etc. This difference can be likened to the difference between imperative programming like C which supports the definion of structs versus the object oriented languages which support richer notions of classes, class hierarchies etc. SPARQL is a query language specially designed to query RDF data sources. Thus, SPARQL is to RDF as XQuery is to XML.

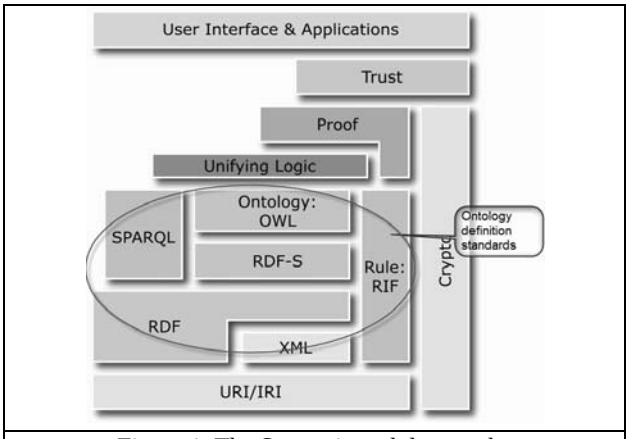

Figure 1. The Semantic-web layer cake

OWL

Although the RDF and the RDFS standards provide a richer set of primitives than XML, the semantic web community found the need for a language that improves RDF by adding constructs that make it more expressive. OWL (Web Ontology Language) is the resultant language which builds on the RDF standards and improves its expressiveness while making sure that the computational complexity of the language is reasonable. By layering it on RDF, many of the RDF tools support and querying can be re-used. The OWL specification itself is designed as three *flavors* (OWL-Lite, OWL-DL and OWL-Full) to support tradeoffs between semantics and computational complexity. OWL-lite is the least expressive dialect of OWL while being simplest to implement and OWL-Full is the most expressive but is also computationally most expensive. OWL-DL falls in the middle of the spectrum of expressiveness and computational complexity. An important functionality that OWL and RDFS support is the ability to derive new information based on the existing information and the schema definition that we refer to as *inferencing*. For instance, we could define a class called *SubsurfaceEntity* and define the class *Well* as a sub-class. When a record asserting "*W001* is a *Well"* is added to the knowledge base, it implicitly adds a record asserting "*W001* is a *SubsurfaceEntity"*. More sophisticated inferencing based on OWL semantics can be used for smart querying of the knowledge base.

Commonly used paradigms for creating data models are relational, ER, UML, XML and semantic web languages (RDF/OWL). Note that the term *data-model* is interchangeably used in literature, to refer to the data-modeling paradigm (relational, XML etc.) as well as the data model instances (also schemas). We have used the word to refer to the latter. The table below summarizes some of the features of ontology languages.

Feature	Category	OWL	RDF/RDFS	XML/XMLS	UML
Classes and class hierarchies	Data modeling	Y	Y	Partial	Y
Properties and Property Hierarchies	Data modeling	Y	Y	N	N
Functional properties (Primary keys), Transitive Properties, inverse properties etc.	Data modeling	Y	N	N	N
Class definition as constraints (E.g. ClosedWell	Data modeling	Y	N	N	N

is a Well that has status=closed)					
Ability to infer new information based on existing information	Reasoning	Y	Y	N	N
Standard representation based on open standards	Representation	Y	Y	Y	Y
Ability to represent and query instance data	Instance Data	Y	Y	Y	Y

OWL satisfies all the key requirements of an ontology language – based on open standards, expressive, and able to store and query instance data. The semantic web standards are emerging technologies and key risks in terms of tool availability and scalability need to be addressed.

3. Applying Ontologies for Oil-Field Operations

As discussed earlier, it is difficult for an engineer trying to make a decision or performing a study, to locate or access a piece of information that he/she needs. It is estimated that a typical petroleum engineer spends upto 60% of his time searching for information [18] . In settings where much of the data is stored in semi-structured or un-structured data-sources (such as simulation models), the problem is further compounded.

Enabling Efficient Access to Simulation Artifacts

Metadata has often been suggested as a way to address this problem of finding information. Metadata is commonly and simplistically defined as data about data. In this work, we have used the term metadata to mean the data that describes the structure and workings of an organization's use of information and the systems it uses to manage that information [3] . Metadata addresses the following five questions: what data do we have, what does it mean, where is it, how did it get there, and how do I get it?[4] . The figure below shows a categorization of metadata types based on the kinds of information it denotes. At the bottom of the metadata types is the syntactic metadata like file size. Structural metadata stores richer metadata like the format of the data. At the higher levels, semantic metadata and ontologies provide the richest description of the data in terms of the meaning of the information within the business context of the organization.

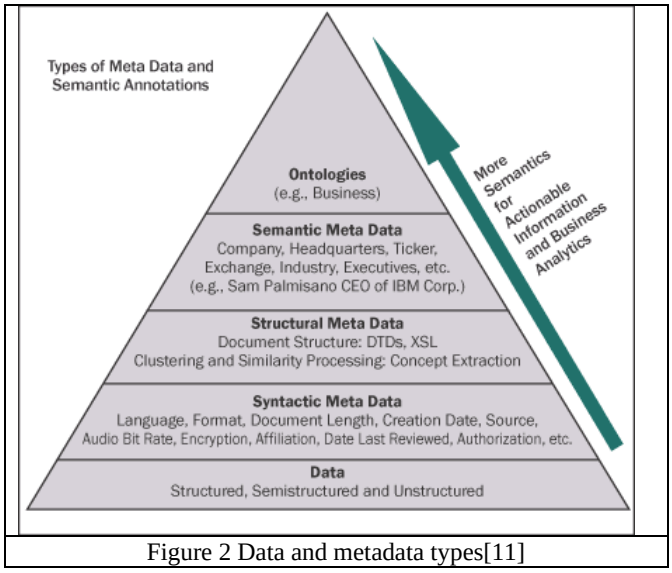

Figure 2 Data and metadata types[11]

The ontology in our solution is centered on the data objects like simulation models, production forecasts,. For each data object, we define three basic kinds of metadata: (i) *access info* metadata which defines how the data object can be accessed, typically its file location, (ii) *provenance* metadata which describes how the data object was created including the name of the person or application that created it, other objects that are related to this object, etc., and (iii) *datatype-specific* metadata that is a concise summarization of the key attributes of the data object that are specific to that particular type of data and are of interest from the domain experts' perspective. For instance, for the *Well* class, the datatype-specific metadata will capture whether the well is a producer or injector, number of completions, zones it produces from, etc. This allows the user to search for data objects which model certain realizations of the asset (e.g., a query such as "Show me all simulation models in which the estimate of the initial oil in place is greater than one billion barrels"). A similar categorization of the metadata elements has been proposed in the grid community [5] . The concepts and relationships, captured in the ontology are further elaborated in Section 4.

The metadata for data objects itself is stored in a knowledge base called the Metadata Catalog. The metadata is created when a engineer publishes a data object into the system. Since much of the metadata is present in the data object itself, special metadata extraction components for each data object were written to parse the data objects and obtain the metadata. This is then persisted in the metadata catalog and consumed by various applications. Some of the tools that use this metadata to provide useful functionality to the end users are presented in Section 5.

Knowledge Management: Tracking Decisions

An engineer making decisions typically performs analyses involving, for example, different uncertainty realizations of the asset and different operational strategies. Much of the information and the rationale behind the decision, including the analyses, is often lost after the decision is made. Such information could be valuable for two reasons. First, an audit trail of the decision making process can be maintained including information about the data consumed and generated during the process, timeline and people involved in that decision, etc. Second, such knowledge capture will contribute towards reliable operation

in face of personnel turnover by enabling new members of the workforce to retrieve and understand in detail the procedures used by predecessors.

Our system allows the engineer to perform analyses by creating different *scenarios*, each of which encodes different operational strategy of the field [6] . Different tools and methods can then be used to evaluate these scenarios and the engineer can choose the scenario which maximizes the target metric. In addition, the analysis is performed under a *decision frame* which explicitly records the setup of the problem, the types of analyses made and omitted, the rationale for doing so, the summary of results from the analyses, etc. For instance, a simulation-based decision frame will record information about the models used for simulation, the output of the simulation, etc. The ontology includes classes and properties that capture this entire process. This information can be queried and retrieved in future for a variety of uses.

4. Ontology Design

In this section we briefly discuss our approach to building ontologies, modularizing them and the salient elements in them. Our ontologies were built for an in-house application, and thus our design goals are different from those of POSC Caesar's Oil and Gas Ontology (OGO), which is intended to be a comprehensive domain vocabulary for the industry [14] . Our ontologies were also designed to be used as a basis for a knowledge-base and for efficiency reasons we have designed it to be small and modular. Currently we have designed three ontologies with approximately 300 classes - the OGO ontology in contrast has more than 10,000 classes. Figure 3 shows the modularization of our ontologies as a set of ontologies that capture different aspect of the problem space. The ontologies in the lower levels of the figure *use* the ontologies in the upper layers. OWL provides the ability to import ontologies defined elsewhere, which makes it easy to modularize ontologies. The elements of the three layers in the figure are described below.

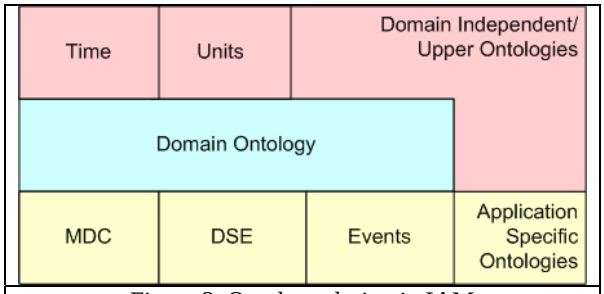

Figure 3. Ontology design in IAM

Domain independent or upper ontologies: which describe concepts which are independent of oil-field semantics. We have considered the (re-)use of time ontology from Pan et al [7] , and the Units ontology from the SWEET ontologies [8] . However, we found that both these ontologies are too detailed for our needs and we hence created smaller subsets of these ontologies to fulfill our needs. Larger ontologies are undesirable because they decrease the performance of reasoning.

Domain ontology: This defines elements of the oil field domain. The elements of the domain ontology are shown in the left hand side of figure 4 below. It mainly consists of physical entities in the oil-field such as Well, Completion, Reservoir, Fault, Zone, etc. Important properties of these entities like initial hydrocarbon estimates for the reservoir, onstream dates for the wells are also captured in the domain ontology. Further important relationships between entities like *drawsFrom* relation which links Well and Region is also captured in the ontology. As can be seen below, the domain ontology itself uses elements of the domain independent ontologies.

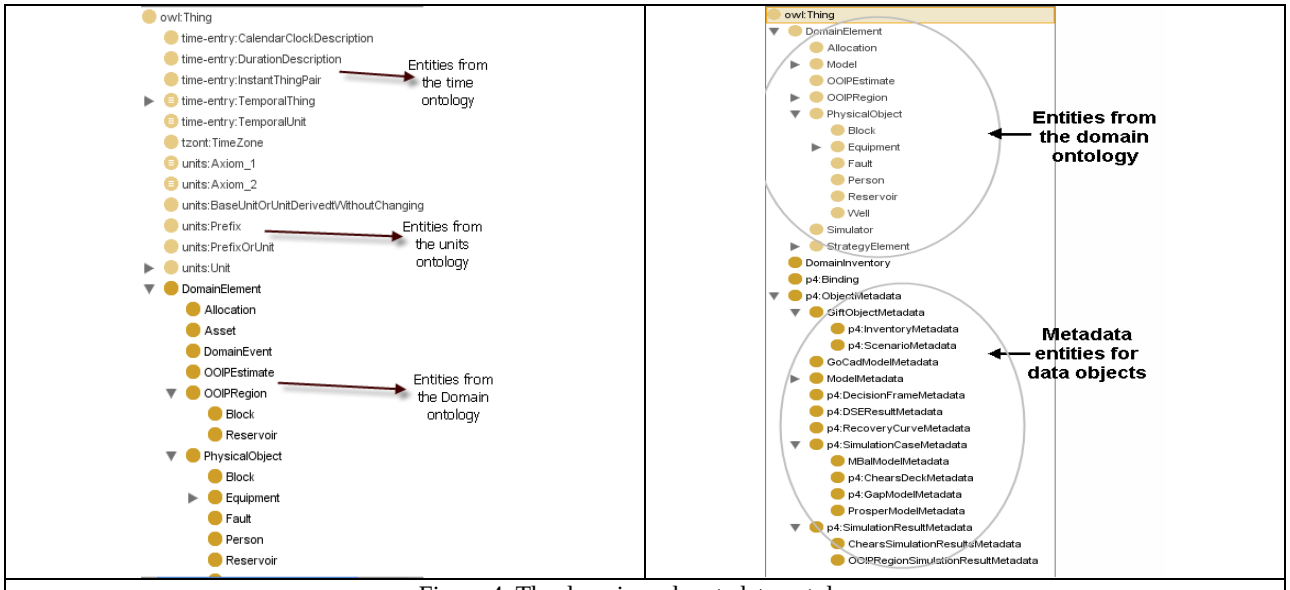

Figure 4. The domain and metadata ontology

The domain ontology is an important artifact in our system and building it has many interesting challenges. Since the domain ontology needs to be application independent i.e. store metadata about the domain elements in the different data stores, it must have minimal *ontological commitment*, i.e., the amount of information in the domain model needs to be minimal. For example, the semantics of an entity called *Well*, is different in different simulation models in the system. The domain model must commit to the least common properties of these applications. When designing a generic application and tool independent domain model, we must address the issue due to the heterogeneities in the way the elements of the domain model are represented in different data sources. Two examples of such heterogeneities mostly commonly observed in our domain are *aliasing*, where the same entity is addressed by different names in different data sources and *scaling* heterogeneities, where different scaling systems (SI vs. Oil field units etc.) are used to encode values. Aliasing is addressed in our system by allowing each entity to have multiple names. To deal with this problem we have created a logical entity called the asset inventory, which contains the physical entities which are the part of the asset. Every model, models one or more of these physical entities and thus a reference from the entities in the model to the entities in the asset inventory is created. Even if an entity is addressed by a different name in different models, since it points to the same entity in the asset inventory, the system understands it to be the same physical entity. The use of an units ontology allows us to address the problem where different data sources used different unit systems to record the information.

Application specific ontologies: We envisage our ontologies to be used in many IAM applications. An example of an application is the metadata store for simulation models. Another example is the Design Space Exploration tool, which enables the engineer to efficiently explore the design space of a problem by using simulations of different granularities. Although these also model elements in the same (oil field) domain, they are different from the domain ontologies because unlike the domain ontologies, the scenarios of usage for these ontologies are restricted to certain applications. Typically application specific ontologies build upon the domain and upper ontologies by using some of the entities described in them. As an example, consider the metadata catalog has its own *application specific* ontology, which is shown in the figure 4 above. As described earlier, each class in the metadata ontology is used to define the metadata associated with each kind of data object, e.g., geological model, reservoir simulation model, network model etc., and the various metadata associated with it. Each data object has metadata summarizing its contents in terms of the entities in the domain ontology. For example, for a simulation model, the domain elements it models and the key properties of these elements are captured.

5. Development Methodology

We have adopted an agile, iterative approach for developing the ontologies. The development was planned in short cycles (sprints), and in each sprint we followed the following steps, similar to those suggested in [12] .

- **Specification**: The domain engineers and the ontologists were involved in defining the applications, e.g., different search parameters, the main parameters for the audit trails etc. A number of subject matter experts from different areas of reservoir engineering were engaged to obtain a broader and a more general view of the problem domain.
- **Conceptualization and Formalization:** Based on the user specifications, the main entities and their attributes were laid out and formalized as OWL axioms. A widely used open source tool called Protégé [10] was used to create the OWL ontologies. Protege allows the designer to create the OWL ontology and saves it in xml based owl representation.
- **Review:** Once the ontologies were created, they were reviewed by the domain experts. The object oriented formalization of the ontologies and the intuitive presentation of the information in Protégé made it easy for the domain engineer to understand and provide feedback, and foster discussion.
- **Application Development:** Application development was carried on in parallel. Due to the iterative software development methodology followed in the project, the ontologies developed during an iteration were passed on to the developers in the next iteration. Constantly changing ontologies could hamper the software development because a change in the ontology could potentially affect the user interfaces, XML schemas used for data transfer, query formats to retrieve data from the knowledge base, etc. Therefore, the modifications were carefully planned and the successive iterations of the ontologies only added elements as far as possible. An automatic code generation package called Jastor [9] was used to create Java code from ontology definitions. This proved to be very useful in coping with the continuously evolving ontologies. We think that the availability of such automated tools will be critical to the adoption and success of this emerging technology.
- **Demonstration:** Finally the applications were demonstrated to the end users for feedback. Not surprisingly, demonstration of functionality and the user interface is a powerful trigger for many ideas and extensions by the user base.

The key stakeholders involved in our development process are:

1. The **business user community** who decide the business entities and relationships are of interest. The key to ontology development is to engage a wide and representative set of business users, playing different roles in the oilfield operation process and with different perspectives on which metadata items and relationships are important enough to be captured in the knowledge base.

2. The **ontology designer** interacts with the business user community to capture the elements of the ontology, and encodes it in OWL. Additionally, it is the job of the ontology designer to communicate the ontology design to the system architect and the engineers on the development side.
3. The **solution architect** designs the overall software solution and manages the development process. He is also responsible for coordinating with the ontology designer and the business user community with respect to the issues related with the ontology – especially the implementation issues that affect or constrain the design of the ontology.
4. The **software developers** build the applications that use the ontologies and thus are the consumers of the ontology.

6. Workflows and Tools

The figure below shows the workflow which describes how the metadata from simulation cases is added to the metadata catalog and queried and used by various IAM applications. When a simulation case is created and validated by an engineer, it is published either through a user interface or by copying it to a agreed upon network location. An IAM agent, which constantly polls this location for new simulation cases then, accesses the model. The metadata is extracted from the simulation model by using *metadata extractors*. Custom metadata extractors are created for each kind of simulation model in the system. We have created metadata components that parse ASCII based documents as well as components that use custom APIs to access information in the simulation models. After the metadata is extracted, reasoning is performed to *materialize* all the inferred information from the information extracted from the model. As mentioned earlier, based on the information (metadata) presented and the OWL ontology definitions, additional information can be *inferred*. In contrast to a materialized knowledge base,, which performs inferencing to create all the new inferred information, when new information is added to the KB, a non-materialized KB derives the inferred data only when a query is issued to it. The advantage of using a materialized KB over a non-materialized one is that the performance of query answering is always faster in the materialized KBs. On the other hand adding information to the materialized KBs can be a slow process and materialized KBs occupy more space. We have adopted this approach in our application because, the frequency of publishing information into our KBs is relatively low when compared to the frequency of queries. After the information is uploaded into the KB, IAM users can access it through the various applications which in turn query the KB and present the information in intuitive ways. Two applications that we have prototyped- the metadata browser and the OOIP tracking application, are discussed below.

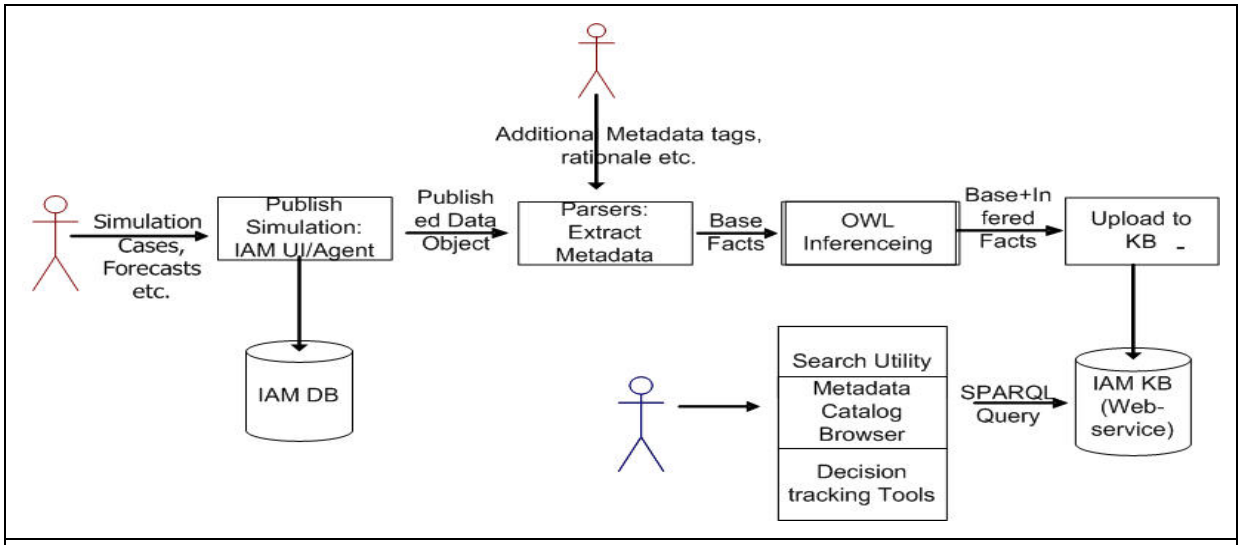

Figure 5. Workflow in the IAM application

Tool 1: Metadata Catalog Browser

The metadata catalog browser allows the user to look at all the data that has been published and search for particular data based on the entities defined in the domain and the metadata ontologies. It also allows the user to search the metadata catalog in intuitive ways. As an example, the user could search for the reservoir model in which the OOIP of the reservoir is greater than a certain number. The user could also search for and navigate through data objects based on its relationships to other data objects. As an example, a search could be to find all the reservoir simulation models which are coarse grained versions of a given model. A screen shot from the tool we developed is shown below.

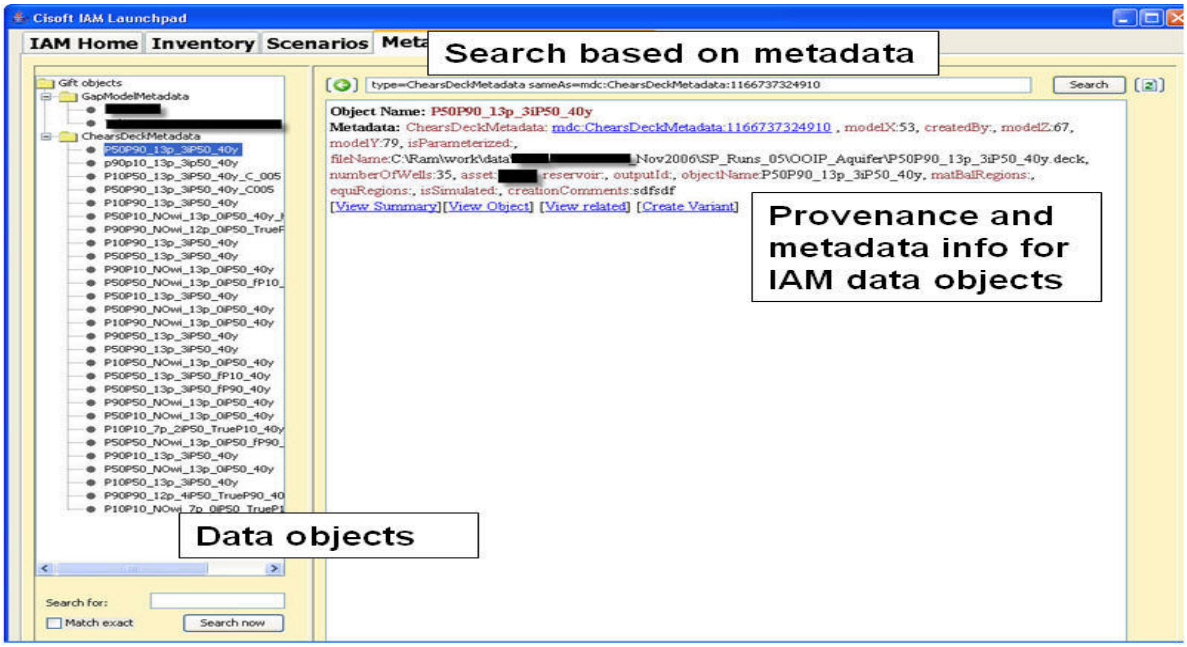

Figure 6. Metadata catalog browser application

Tool 2: OOIP Comparison utility

An important uncertainty parameter that is used in different kinds of models like the geological model, reservoir model and the field model is the OOIP information of the whole field as well as different regions of the field. A tool that allows the geologists or the reservoir engineer to compare the assumptions about the OOIP values in each of the models is extremely invaluable and provides insight into the understanding of the asset by the other engineers. The screenshot below shows a tool that provides such functionality. The regions in one model are not always the same as those modeled in another and thus a way to *map* the regions is provided, so that the user can compare regions in different models. This application also provides a unified view of the information across models, because the estimates in different models can be compared irrespective of the various heterogeneities across the models.

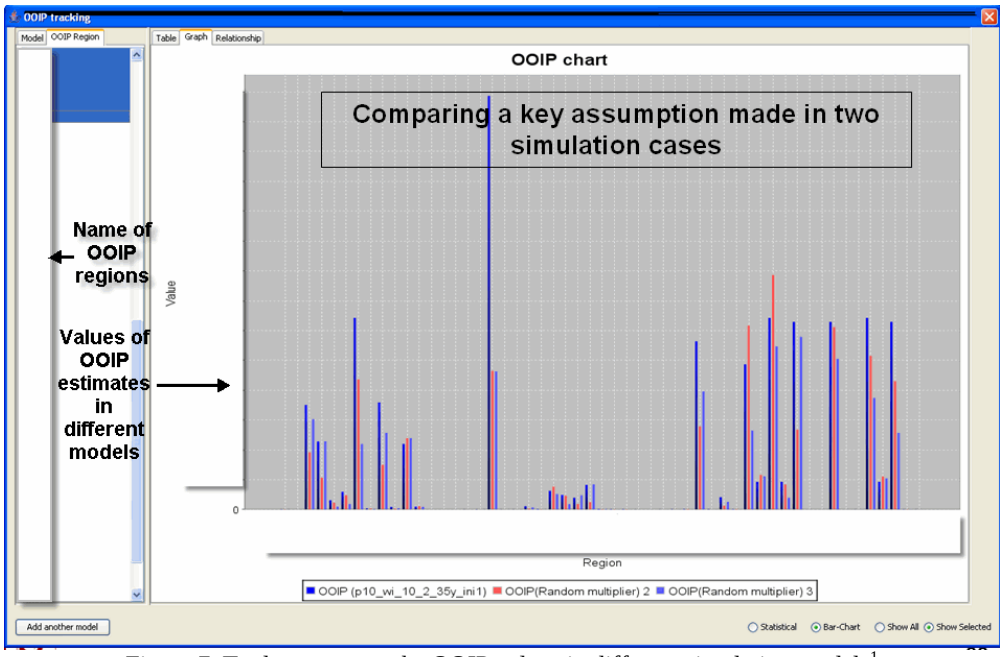

Figure 7. Tool to compare the OOIP values in different simulation models[1]

[1] Names of regions and their OOIP estimates are intentionally obfuscated.

7. Discussion

An important reason for choosing the semantic web technologies was because of our strategic vision of IAM as a metadata and knowledge management platform for oilfield development and operations domain. We believe that the expressiveness afforded by the semantic web stack will prove to be highly beneficial for such functionality. As the semantic web technologies are still emerging, we are addressing many issues with respect to the scalability, performance and tool support. Early performance testing was performed by using synthetic but realistic OWL data sets of various sizes, combined with different simulated user loads in terms of metacatalog queries. Although the performance appears to be acceptable for relatively small data sets, more foundational research is warranted to improving the performance and scalability of OWL knowledge bases. In particular, the OWL inference capabilities are the slowest and most resource intensive in terms of processor and memory usage. We are investigating various methods to improve the performance and scalability of OWL inferencing. With major vendors like Oracle announcing support for these technologies in their product offerings [13] , we are hopeful of dramatic improvements in these areas in the near future.

Another key issue encountered while building our solution was the limited number of tools supporting scalable, industry-strength deployment of OWL-based systems. Although our development platform of choice was Microsoft .NET, we could not find a toolkit in the .NET framework that provided OWL support. Our entire OWL specific code (the Metadata catalog) was coded in Java using an open source framework called Jena [20]. By abstracting the metadata catalog as a web-service, we were able to use our preferred platform for developing the rest of the solution including the metadata extractors and the client applications. More ubiquitous tool support will be necessary for the success and wide spread adoption of this technology in the enterprise.

When designing an ontology in OWL, the designer must be cognizant of the constructs being used and whether the reasoning tools that will be used support the OWL features used in the ontology. For example, many of the currently developed OWL reasoning tools like Jena and Oracle are implemented using rule based engines, which only implement a subset of the OWL-Lite semantics. Constructs like nominals are not supported in any of the tools. Of the freely available reasoning only Pellet supports all the OWL features, but we have found the performance of such tableaux methods to not be satisfactory for our data-sets.

8. Acknowledgements

This research was funded by CiSoft, (Center for Interactive Smart Oilfield Technologies), a Center of Research Excellence & Academic Training and a joint venture between the University of Southern California & Chevron. We are grateful to the management of CiSoft and Chevron for permission to present this work. We are thankful to Pradeep Kumar and Laurent Pianelo (Chevron) for significantly contributing to the definition of the ontology, and providing feedback on the prototypes. We acknowledge the contributions of Kanwal Gupta (Avanade) to the design of the IAM system architecture.

9. Reference

[1] Tim Berners-Lee, Semantic Web Roadmap, http://www.w3.org/DesignIssues/Semantic.html
[2] Wikipedia Entry , Ontology, http://en.wikipedia.org/wiki/Ontology_%28computer_science%29
[3] Hay, D.C.: Data Model Patterns: A Metadata Map. Morgan Kaufmann (2006)
[4] Tannenbaum, A.: Metadata solutions: using metamodels, repositories, XML, and enterprise portals to generate information on demand. Addison-Wesley (2002)
[5] Singh, G., Bharathi, S., Chervenak, A., Deelman, E., Kesselman, C., Manohar, M., Patil, S., Pearlman, L.: A metadata catalog service for data intensive applications. In: ACM/IEEE conference on Supercomputing. (2003)
[6] Cong Zhang, Viktor Prasanna, Abdollah Orangi, Will Da Sie, Aditya Kwatra, Modeling methodology for application development in petroleum industry, IEEE International Conference on Information Reuse and Integration, Las Vegas, 2005.
[7] Hobbs, J.R., Pan, F.: Time ontology in owl.ontology engineering patterns task force of the semantic web best practices and deployment working group, world wide web consortium (W3C) notes.
[8] Sweet ontologies: Semantic web for earth and environmental technologies
[9] Jastor: http://jastor.sourceforge.net/
[10] Protégé ontology editor and knowledge acquisition system: http://protege.stanford.edu/
[11] A. Sheth, Semantic Meta Data for Enterprise Information Integration, http://www.dmreview.com/issues/20030701/6962-1.html
[12] Pinto, H. S. and Martins, J. P. (2004). Ontologies: How can they be built? *Knowledge and Information Systems*, V6(4):441-464.
[13] Oracle Semantic Technologies Center, http://www.oracle.com/technology/tech/semantic_technologies/index.html
[14] POSC Caesar Association, http://www.posccaesar.com/
[15] Frank Chum, "Use Case: Ontology-Driven Information Integration and Delivery A Survey of Semantic Web Technology in the Oil and Gas Industry," April 2007. http://www.w3.org/2001/sw/sweo/public/UseCases/Chevron/
[16] David Norheim and Roar Fjellheim, "AKSIO - Active Knowledge management in the petroleum industry", 3rd European Semantic Web Conference (Industry Forum), June 2006.
[17] Cisoft IAM project, http://pgroup.usc.edu/iam
[18] Lesslar, P.C., van den Berg, F.G., Sarawak Shell Berhad, Managing Data Assets to Improve Business Performance, SPE Asia Pacific Conference on Integrated Modelling for Asset Management, 23-24 March 1998, Kuala Lumpur, Malaysia
[19] Cong Zhang, Amol Bakshi, Will DaSie, V.K. Prasanna, Birlie Bourgeois, A Framework for Design Space Exploration in Oilfield Asset Development, (To appear) SPE Intelligent Energy Conference, February 2008 (SPE 112268).
[20] Jena- A Semantic Web Framework, http://jena.sourceforge.net/