SPE-128708

Semantic Web Technologies for Event Modeling and Analysis: A Well Surveillance Use Case

Tao Zhu, Univ. of Southern California; Amol Bakshi, SPE, Chevron; Viktor K. Prasanna, SPE, Univ. of Southern California; Roger Cutler, Chevron; Scott Fanty, Chevron

Abstract

The ability to identify events of interest at the surface and sub-surface, and use event information for various workflows and analyses is a key enabler for a variety of surveillance, monitoring, and optimization workflows. The scalability, extensibility, and maintainability of an event management system is directly related to the conceptual model used to represent events, the ease with which relationships between events can be encoded in the form of rules, and the type of rule engine that performs the analysis necessary to extract useful, actionable information from the frequently updated event database. This paper summarizes our experience with building a well surveillance application using semantic web technologies. Events are modeled using the web ontology language (OWL) and relationships between events expressed as rules in the semantic web rule language SWRL. An off-the-shelf rule engine aggregates the values of lower level attributes related to well performance and not only infers the well status ("alert" or "no-alert") but also provides the entire chain of reasoning that led to the particular well status. Although more work needs to be done to translate the reasoning into a form that is comprehensible to the domain experts, even the raw information is a valuable aid in understanding and validating the rule base. We discuss the advantages of using semweb technologies for event management: the relatively low code complexity, the ability to remove existing rules or add new rules of arbitrary complexity at run time, and the ease of encoding domain knowledge into events and rules. Evaluation of time performance of our application for different number of wells is presented. The broader significance of this work is in the context of understanding the right technology (or combination of technologies) to build logical, extensible, and maintainable systems for processing field events.

1. Introduction

Event information is valuable to a number of workflows related to surveillance, monitoring, and optimization. Important attributes of an event include its type, time of occurrence, the entity affected by the event, the status of the entity before, during, and after the event, and the relationship of that event to other events. For example, detection and diagnosis of problem wells could rely on analysis of well events such as recent well work and status changes (e.g. production rate, efficiency of lift equipment, etc.). Collaborative, cross-domain analysis of events is hindered by the fact that event information is typically distributed among various sources and in different formats, and there is no common representation of domain knowledge related to event relationships. Traditional approaches to event analysis involve tedious information gathering and integration, and manual interpretation of event information. While this methodology might be adequate for localized analysis of simple events, it will not scale to situations that require analysis of large amounts of event information from a multitude of sources.

In this paper, we explore the utility and applicability of semantic web technologies [Berners-Lee2001] to build an event modeling and analysis system. We designed an event model using the Web Ontology Language (OWL) [Bechhofer2004] to uniformly represent events as well as their features and inter-relationships. Based on the event model, we use the Semantic Web Rule Language (SWRL) [Horrocks2004] to represent knowledge about event relationships. In our approach, relationships between events are automatically inferred by using an off-the-shelf rule engine. The purpose of our work is to explore the semantic web technologies as a potentially promising approach to building extensible and maintainable systems for processing field events.

Our case study is based on the Well Event Surveillance Tool (WEST) [Schipperijn2009]. The objective was to understand if we could design a system that could provide a similar well surveillance functionality from the users' perspective as provided by WEST, but using a different underlying technology. In addition, we demonstrate that in our approach, the derivation history provided by the rule engine allows the users to understand not only the end result of the event processing (e.g., that the status of "Well A" is critical) but also the chain of inferencing that led to that inference. We also evaluate the time performance of our implementations and some application specific tuning techniques for performance optimization.

The rest of the paper is organized as follows. In Section 2 we introduce some background knowledge about semantic web technologies. Then we sketch our event management framework in Section 3. In Section 4 we introduce the use case of exception driven well surveillance and the WEST tool. In Section 5 we describe in detail the implementation of our approach using the well surveillance use case. Section 6 presents results of performance evaluation. We summarize the advantages of our approach in Section 7.

2. Background

   Semantic Web: The semantic web [Berners-Lee2001] is an evolving extension of the World Wide Web in which the semantics of information is well defined to be both human understandable and machine processable. It envisions the usage of the web as a universal medium for data, information, and knowledge exchange. To achieve this vision, the W3C (World Wide Web Consortium) has proposed a set of standards and enabling technologies to process the semantics of information. Some key specifications include RDF (Resource Description Framework), RDFS (RDF Schema), and OWL (Web Ontology Language). Although originally proposed for the web, semantic web technologies have been applied to information management in domains including healthcare and life sciences [Denney2009, Kawazoe2008] and the oil and gas industry [Feigenbaum2008, Norheim2006, Soma2007].

RDF: RDF (Resource Description Framework) [Manola2004] is a general method for conception modeling of information, based on the idea of making statements about resources in the form of subject-predicate-object triples. The subject represents the resource, and the predicate denotes properties of the resource and expresses a relationship between the subject and the object. For example, in the triple <w1, domain:hasName, "INJ001">, "w1" is a resource, "domain:hasName" is a property ("domain" is the namespace of the property), "INJ001" is a string. This triple means that "a resource w1 has a name of INJ001". Together with another triple <w1, rdf:type domain:Well>, it represents the fact that "there is a well called INJ001". RDFS (RDF Schema) is an extensible knowledge representation language to define RDF vocabularies which are used to structure RDF resources. A query language called SPARQL is proposed to query RDF data.

OWL: OWL (Web Ontology Language) [Bechhofer2004] is a family of knowledge representation languages endorsed by the W3C for authoring ontologies. An ontology is a formal representation of a set of concepts in a domain and the relationships between these concepts. It provides a shared vocabulary which can be used to model a domain.

SWRL: SWRL (Semantic Web Rule Language) [Horrocks2004] is a proposed rule language for semantic web. A SWRL rule consists of an antecedent (rule body) and a consequent (rule head), each of which consists of a set of atoms. The meaning of a rule is that if the antecedent is true, the consequent must also be true. SWRL rules are in Horn-like format, which means the antecedent is a conjunction of atoms while the consequent contains a single atom. For example, in the rule "parent (?x, ?y) Λ brother (?y, ?z) $\rightarrow$ uncle (?x, ?z)", the antecedent contains two atoms "parent (?x, ?y)" and "brother (?y, ?z)", the consequent has a single atom "uncle (?x, ?z)". In the atom "parent (?x, ?y)", ?x and ?y are variables, "parent" is the relationship between these two variables (represented by a property in OWL), which says that ?x 's parent is ?y. An atom can also be unary, which represents an "is-a" relationship (represented by a class in OWL). For example, person (?a) means that ?a is a person.

3. Our Framework
   There are three major components in our event management framework: the event ontology, the rule based event correlation, and the user interface. The *event ontology* defines a data model for events using the OWL language. It represents a common understanding of the semantics of events, their features and relationships between each other. The event ontology acts as a schema for event data, based on which events are represented and stored in a uniform format. The *rule based event correlation* module is responsible for applying rules of event relationships to the event instance data and automatically infer relationships between events. An off-the-shelf rule engine is employed, from which the derivations of inferred facts can be traced. A derivation of an inferred fact is the chain of rules and facts that lead to the inferred fact. It is in the format of a directed graph. The derivation of an inferred fact tells where the fact was generated from, which means in our problem that we can not only get an implicit event

relationship, but also why there is such a relationship. This makes our approach go a step further towards event analysis rather than just event surveillance. The *user interface* provides interfaces for the user to browse and query for event information, including event instances, inferred event relationships, and the derivations of inferred relationships. The requested information is visualized through the user interface. It also allows the user to create and edit the rules to incorporate their domain knowledge of event relationships.
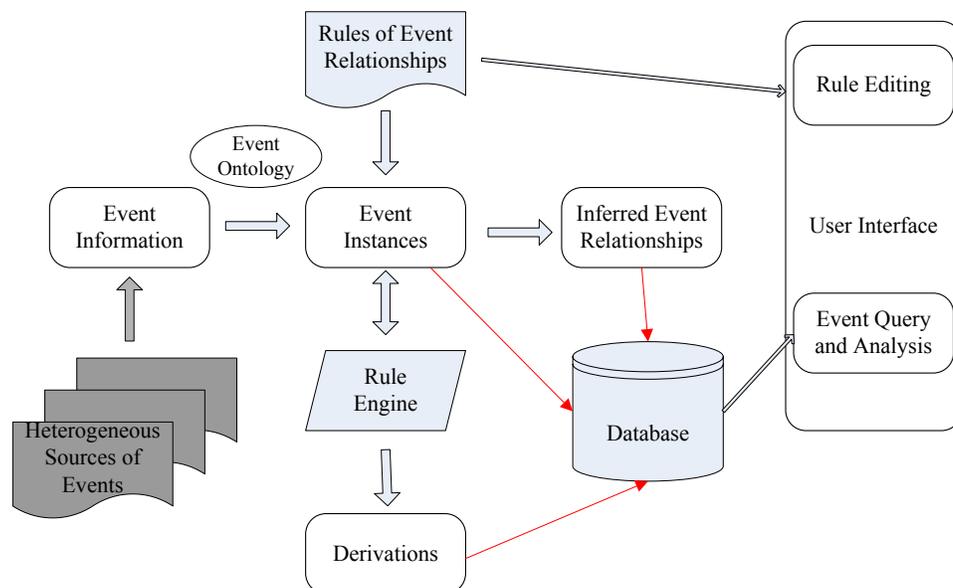
Figure 1. Architecture of the event management framework

As shown in Figure 1, event information is captured from heterogeneous sources including spreadsheets, databases, logs generated by software applications, etc. This information is used to populate event instances (in the format of RDF/OWL triples) based on the event ontology. Rules defined in SWRL are applied to these instances using an off-the-shelf rule engine to infer relationships between events. Derivations of the inferred facts are traced from the rule engine and are stored in a database together with the inferred relationships and the event instances for querying. A user interface is built for rule creation and editing, as well as querying and visualization of event information.

4.   The Well Surveillance Use Case

   The objective of well surveillance is to monitor and identify underperforming wells while minimizing downtime. This process is usually started by collecting data from multiple sources and then manually analyzing the information in an attempt to identify and diagnose problems. Based on the analysis and diagnosis, decisions about the underperforming well could be made. This approach is time consuming and far from optimal.

The WEST tool [Schipperijn2009] seeks to improve the efficiency of well surveillance by automating manual processes. WEST defines a set of attributes to represent the status of a well. The values of the attributes are computed daily from well data in a database. These values are then mapped to ranges such as "Low", "Moderate", and "High". An expert system is built to determine the surveillance status of a well based on the classified attribute values. For example, if the pump efficiency is "Low" and the fluid level is "High", the pump status is "Poor". This is done by matching the attributes' values with a set of decoding keys which represents the mapping between attributes to surveillance status.

To reduce the number of decoding keys to match, WEST develops a hierarchical architecture, which first aggregates attributes to a set of classes, and then determine the surveillance status by examining the values of the classes. For example, the attributes of "pump efficiency" and "fluid level" belong to a class called "pump status". The values of "pump efficiency" (Low) and "fluid level" (High) are matched with a decoding key which tells the value of the "pump status" (Poor) based on the values of its two attributes. The value of the class status "pump status" is then used along with other classes' values to determine the well surveillance status.

WEST also introduces the concept of "logic round" to represent the prioritization between groups of classes when they are considered for exception detection. A logic round consists of a group of classes, which are evaluated simultaneously in this logic round. The logic rounds are evaluated in primary order beginning with the first. If the result of the evaluation of a logic round is "null", the next round is evaluated. Otherwise the surveillance status of the well is the result of the current logic round. Such a mechanism of prioritization also helps reduce the number of decoding keys to maintain in the system, which improves the efficiency.

We developed our event management framework using semantic web technologies for modeling the event instances and the relationships between the instances. As mentioned earlier, the primary objective was to understand if we could use a different underlying technology to provide similar well surveillance functionality from the users' perspective as provided by WEST. A secondary and equally important goal was to understand the strengths and limitations of using semantic web technologies for modeling events and rules, and apply the learnings to similar systems that could aggregate and analyze events across the surface and subsurface, and at different levels of abstraction.

5. Implementation

5.1. Event ontology

We built our event ontology in a modular way, starting with importing the OWL-Time ontology [Hobbs2004] into our event ontology to describe the temporal feature of an event. The OWL-Time ontology defines a set of time concepts, such as time point, time interval, etc., as well as relationships between different temporal instances, such as "before", "after". To model the relationship between an event and a field entity (e.g. a well, a pump), we imported a domain ontology [Soma2007] that defines basic concepts and relationships among key oilfield entities such as reservoirs, zones, completions, wells, and surface facilities. Such a modular design allows our event ontology to focus on defining event-related concepts and leverage existing ontologies from the domain.
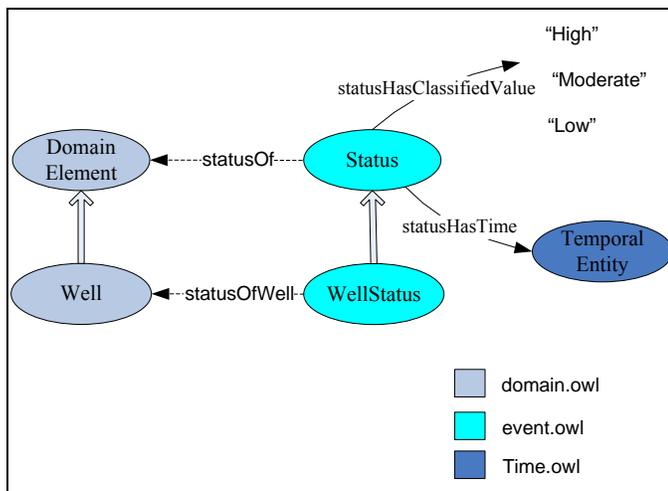
Figure 2. Core Event Ontology

Figure 2 shows a subset of the classes and properties defined in the core event ontology. Class "Well" is a subclass of "DomainElement", both of which are defined in the domain ontology. The class "Status" defined in the event ontology represents the status of a domain entity such as a well, thus it has a property called "statusOf" pointing to the "DomainElement" class, meaning that it is the status of a specific domain entity. "WellStatus" is a subclass of "Status" which specifically represents status of a well, having a property "statusOfWell" pointing to the "Well" class. (Note that "statusOfWell" is a sub-property of "statusOf".) As the dynamic domain status changes over time, the "Status" class has a property called "statusHasTime" to denote the valid time of this status. The range of the property "statusHasTime" is the class "TemporalEntity" defined in the OWL-Time ontology, which is further inherited to be either a time point (instant time) or a time period. In the WEST use case, the values of a well status (attribute) are classified into different ranges, so we create a property for the "Status" class called "statusHasClassifiedValue" whose range is a collection of possible classified values of the attributes. (Note that a subclass automatically inherits the properties of its parent class, so the "WellStatus" class also has the properties "statusHasClassifiedValue" and "statusHasTime".)

The "WellStatus" class is extended to subclasses of production well statuses and injection well statuses (Figure 3). We model every attribute or class defined in the WEST tool as a subclass of either "ProductionWellStatus" or "SteamInjectionWellStatus". For the convenience of description, we assume the possible classified values of all attributes and classes are "Low", "Moderate" and "High". It is easy to add other possible values such as "Poor", "OK" to the property "statusHasClassifiedValue" in the event ontology. If necessary, one can also create a sub-property of "statusHasClassifiedValue" for a specific subclass of "WellStatus" to customize its possible classified values.
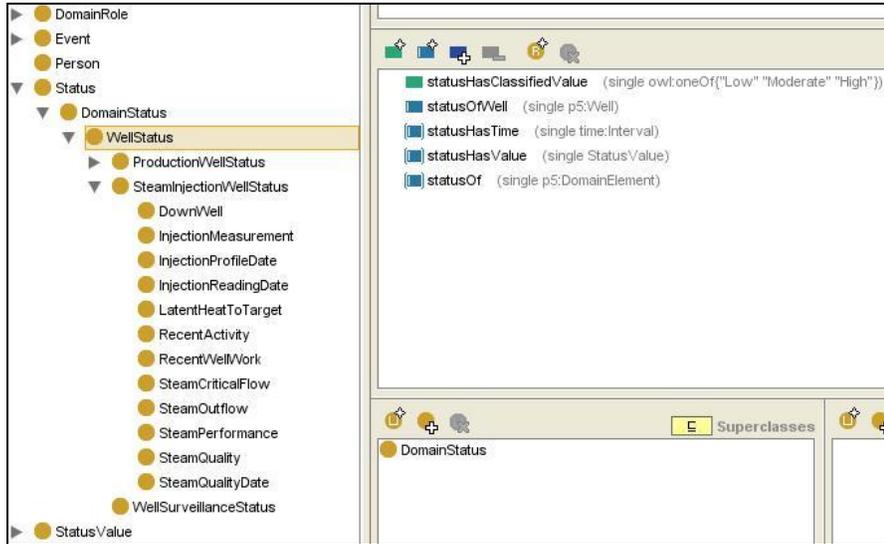

Figure 3. Classes and properties defined in the ontology

Instances are created based on the event ontology to record event information. For example, the fact that "the value of well W01's status 'pump efficiency' is 'Low'" is represented as triples <?s, rdf:type PumpEfficiency>, <?s statusOfWell W01>, <?s statusHasClassifiedValue 'Low'> (there is also a triple in the domain ontology <W01, rdf:type Well> which means that "W01" is a well).

5.2. Rule based reasoning for exception detection

SWRL rules are defined to represent the mappings from attributes to classes, and from classes to well surveillance status. Such mappings are maintained as decoding keys in WEST. For example, the decoding key which says that "if the pump efficiency of a well is 'Low', and the measurement quality of the injection well is 'High', then the pump status of the well is 'High' (Alert)" can be represented by a SWRL rule defined based on our event ontology:

PumpEfficiency (?p) Λ statusOfWell (?p, ?w) Λ statusHasClassifiedValue (?p, 'Low') Λ InjectionMeasurement (?q) Λ statusOfWell (?q, ?w) Λ statusHasClassifiedValue (?q, 'High') Λ PumpStatus (?s) Λ statusOfWell (?s, ?w) → statusHasClassifiedValue (?s, 'High').

Similarly, every decoding key that maps the values of a set of attributes to the value of a class, or values of classes to the surveillance status, can be represented by a SWRL rule.

We can also express "OR" and "NOT" in the rule conditions using SWRL rules. Although SWRL requires the relationship between axioms in the rule body to be "AND", we can describe the "OR" relationship by breaking the rule into multiple rules. For example, the rule ["DownWell" is "High" OR "WellWork" is "Low" → "RecentActivity" is "High"] can be written as two rules

DownWell (?a) Λ statusOfWell(?a, ?w) Λ statusHasClassifiedValue (?a, 'High') Λ RecentActivity (?b) Λ statusOfWell(?b, ?w) → statusHasClassifiedValue (?b, 'High')

and

WellWork (?a) Λ statusOfWell(?a, ?w) Λ statusHasClassifiedValue (?a, 'Low') Λ RecentActivity (?b) Λ statusOfWell(?b, ?w)
→ *statusHasClassifiedValue (?b, 'High')*

For the "NOT" operator, SWRL supports the built-in function "notEqual" to express the negation of certain value, for example, notEqual(?v, "Low") means the value of the variable "?v" is not "Low".

We can also use SWRL rules to express prioritization as the logic rounds in WEST. We illustrate this by using an example, in which we consider two logic rounds: the first logic round contains one class "RecentActivity", while the second logic round contains one class "InjectionMeasurement". In other words, the value of the "RecentActivity" status of the well is first checked. If the value of this attribute is "High", then the well surveillance status is inferred to be "High" (Alert), otherwise the surveillance status is determined by also looking at the value of "InjectionMeasurement". This can be expressed using two rules:

RecentActivity (?a) Λ statusOfWell(?a, ?w) Λ statusHasClassifiedValue (?a, 'High') Λ WellSurveillanceStatus (?b) Λ statusOfWell(?b, ?w) → *statusHasClassifiedValue (?b, 'High')*

and

RecentActivity (?a) Λ statusOfWell(?a, ?w) Λ statusHasClassifiedValue (?a, ?v) Λ notEqual(?v, 'High') Λ InjectionMeasurement (?b) Λ statusOfWell(?b, ?w) Λ statusHasClassifiedValue(?b, ?v2) Λ WellSurveillanceStatus (?b) Λ statusOfWell(?b, ?w) → *statusHasClassifiedValue (?b, ?v2)*

5.3. User interfaces

A basic visualization of well status and derivations is built for users to browse the wells to view the values of the attributes representing the well status (Figure 4). When the user expands the "Injection Wells" or "Production Wells" in the browser panel of well status, a list of wells is displayed, which is populated by querying the event ontology. Each well item contains information of the well's name and its surveillance status . When each well item is further expanded, a tree of well status is shown. As in WEST the attributes of well status are organized in a hierarchical way, we display the well status in the format of a tree. Specifically, the parent nodes represent the classes in WEST, and the children nodes represent attributes in each class. Similarly, each item in the tree of attributes contains information of the name of the attribute and its value ("Low", "Moderate", "High").
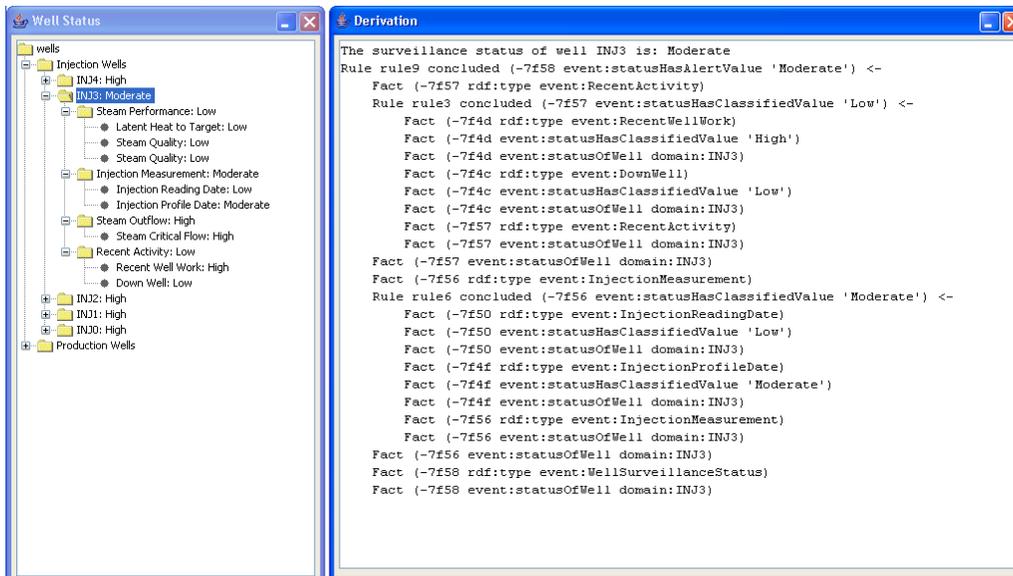

Figure 4. Well status and derivations

When an item of well is selected, the derivation of its surveillance status is shown (the panel on the right in Figure 4). The derivation of a particular fact (e.g., that a particular well has an "Alert" status) is usually in the form of a directed graph that represents the entire chain of reasoning used to arrive at that conclusion. We built a basic visualization of the derivation which prints out the raw information of the facts and rules in the derivation traced from the rule engine. As shown in Figure 4, the

inferred fact (the surveillance status of a well) is printed first, followed by the rule that inferred this fact, and the list of facts that fired this rule. If one of these facts that fired the rule is also an inferred fact, then its derivation information is also printed out, and so on. The rule conditions and facts are printed in the format of triples (<id of the resource, property, value of the property>).

The example in Figure 4 shows the surveillance status of the injection well INJ003 is "Moderate" (possibly means "pending alert"). The derivation information tells us that its status is "Moderate" because its "Recent Activity" is "Low" (Normal) and its "Injection Measurement" is "Moderate", which fired "rule9" saying that if a well's "Recently Activity" is "Low", then the surveillance status is determined by its "Injection Measurement" (Recall the example of logic round in section 5.2). A more user friendly visualization can be built by depicting this chain of reasoning as a directed graph.

The user interface also includes a dialog for users to view and edit the rules which represent the domain knowledge of event relationships (Figure 5). The dialog contains a list of defined rules, the prefix of the rule set which defines namespaces used in the rule definition and editable fields for writing the rules and comments for each rule. While we are using the Jena rule based reasoner as the rule engine, currently the user interface requires rules to be written in the Jena rule language. An opportunity for future work is the design of an interface that will be usable by domain experts who are not versed in the Jena rule language.
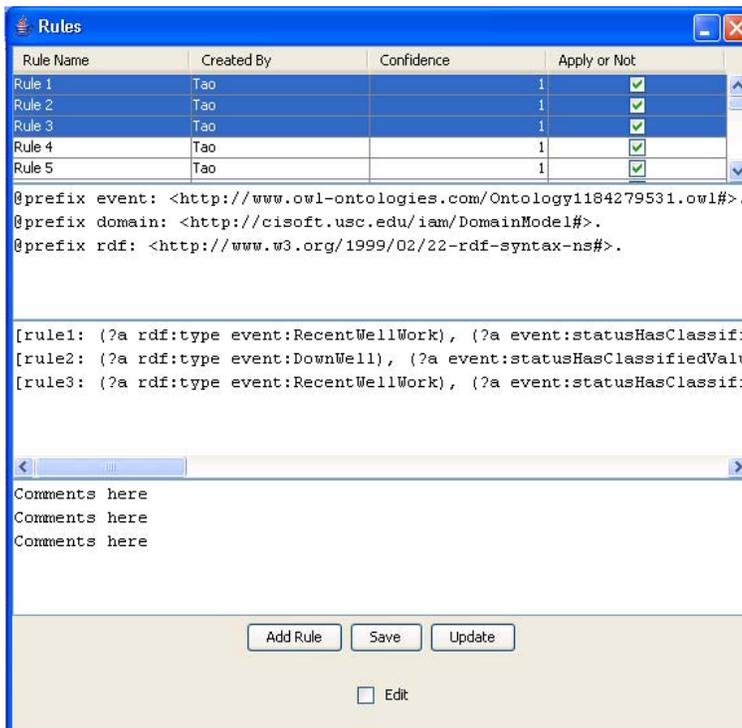

Figure 5. Rule editing

6.   Performance Evaluation and Tuning

To evaluate the feasibility of our approach, we measured the time performance of our implementation of the well surveillance use case for different numbers of wells. The hardware of our testing environment is a workstation with a CPU of Quad-Core AMD 2.01G and 32GB memory.

We create synthetic rules for injection and production wells using the attributes defined in WEST - 24 rules for injection wells and 144 rules for production wells. Similar to the decoding keys in WEST, the number of rules depends on the number of attributes, classes, the number of attributes in each class, and the setting of logic rounds. We randomly generated attribute values for different wells and used them to populate instances of the well status classes in the event ontology. In the real application, the values of the attributes will be computed from well data in the appropriate system of record. We apply the rules to the instances of

all wells at the same time, and measure the total time cost of loading the instances of well status, inferring the well surveillance status and querying for the inferred well surveillance status and derivations, for different numbers of wells.
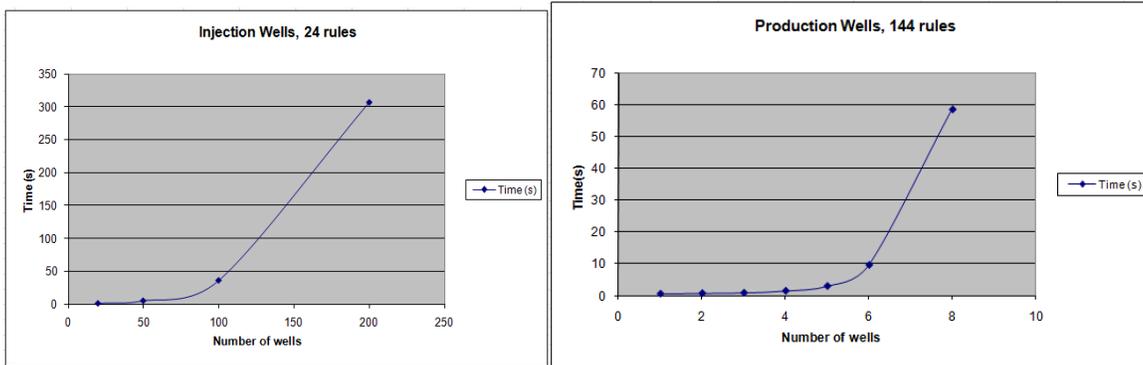

Figure 6. Time cost for different number of wells

As we can see from the results in Figure 6, the time cost increases dramatically with the number of wells. It takes nearly an hour to process 500 injection wells, and about 4 hours for 500 production wells. While there are opportunities to speed up the inferencing algorithm by using parallel processing techniques [Soma2008] in the rule engine, we can also exploit the "embarrassingly parallel" nature of rule processing in this use case.

The time performance is mainly dominated by the time cost of the rule based reasoning process. When the number of wells increases, the number of instances in the ontology increases, which slows down the inferencing process quickly, because it requires more memory to store the instances and more time to check the instances when trying to match a rule's conditions. So an important issue of performance tuning is how to reduce the number of instances that are fed to the rules at a time. In the WEST workflow, a well's surveillance status (Alert or No-Alert) is determined by checking the attribute values for that well only – i.e., the status of a well does not depend on the status of any other well. Per this assumption, wells can be evaluated independently, i.e. we can apply rules to the instances of one well (or multiple wells as a group) to get the inferred surveillance status of this well (or well group), and finally aggregate the surveillance status of all wells, without harming the correctness of the inferencing process.

To improve performance, we divide the instances of well status into multiple batches, each of which contain instances of one or multiple wells. Then we apply rules to different batches independently and sequentially. Finally we aggregate the inferred results of different batches to get the surveillance status of all wells. The result of the time performance is shown in Figure 7. The time cost includes the time of loading all the batches, total time cost of processing the batches (applying rules to different batches sequentially), and the time of querying the surveillance status of all wells and their derivations.
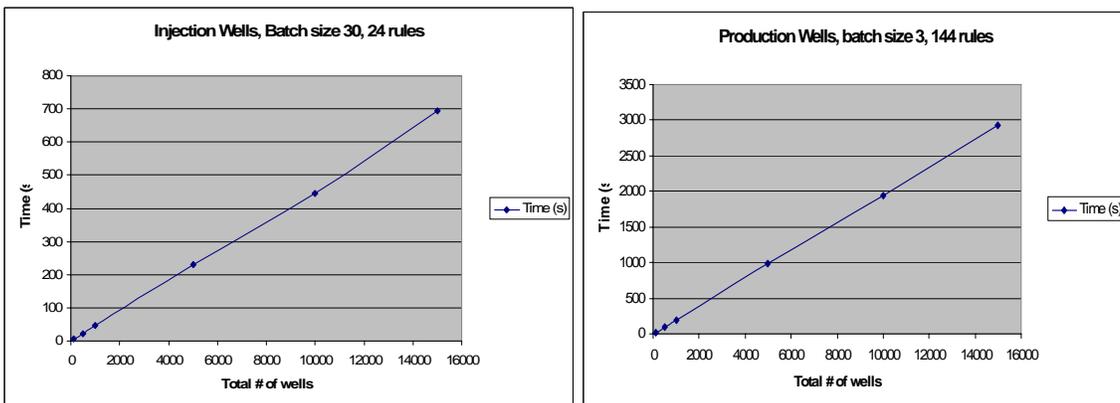

Figure 7. Time cost using batch processing

As the results in Figure 7 suggest, the time cost now increases linearly with the number of wells. This is because the time cost of processing a batch (instances of one or multiple wells) is almost fixed as the numbers of instances in different batches are the same. While we process the batches in a sequential way, the time cost should be the time cost of processing a single batch

multiplied by the number of batches. It costs less than 12 minutes to process 15,000 injection wells, and less than 50 minutes to process 15,000 production wells.
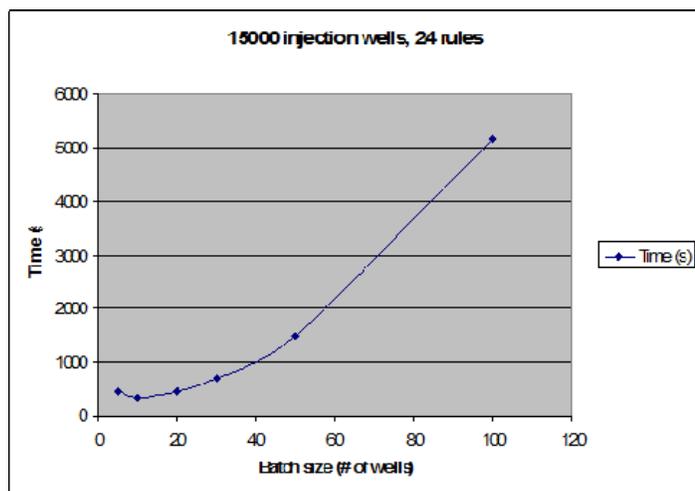

Figure 8. Influence of batch size on the overall time performance

We also study the influence of the size of a batch to the time performance (Figure 8). While the time cost of applying rules to the instances increases much faster than the number of instances, it seems better to have more batches (fewer instances in a single batch). However, as the number of batches increases, the overhead of loading the batches when performing inferencing and querying well status is higher, because the time cost of initialization in loading and querying is higher. So there should be an optimal batch size which makes the overall time cost minimal.

From Figure 8 we can see that for the case of 15,000 injection wells, the optimal batch size is about 10 wells. In real problem settings, the optimal batch size depends on the number of rules, number of instances of well status per well and the total number of wells.

After tuning the implementation for performance optimization, the time performance is quite acceptable for a reasonably large scale of wells. Although the assumption of independence may not hold for all scenarios, we can divide the set of instances into independent subsets in different granularity such as well groups, locations, assets, time periods, etc. based on real problem settings.

7. Discussion

We have shown the feasibility of applying semantic web technologies to event management in reservoir engineering by using it to implement functionality that is equivalent to the WEST tool [Schipperijn2009] for exception driven well surveillance. We defined an OWL ontology to model the events and well status, represented the logic of detecting exception/alerts from well events and status using SWRL rules, and employed an off-the-shelf rule engine to automatically determine the surveillance status of wells. We demonstrated that our approach not only provided comparable functionalities as WEST, but also provided users with the derivation information of well surveillance status. OWL, which is based on description logic, is capable of describing the events and status in well surveillance. The rule language SWRL is expressive enough to representing the classification rules (decoding keys). Our performance testing shows that the time performance is acceptable in a daily surveillance detection scenario (approx. 12 minutes to process 15,000 injection wells with 24 rules, and approx. 50 minutes to process 15,000 production wells with 144 rules). By using batch processing, the time performance scaled well (linearly) with the number of wells.

The advantages of semantic web technologies for this use case can be summarized as follows:

Intuitive representation: Modeling an event in terms of a set of statements about that event and relationships to domain entities is more intuitive from a domain expert's perspective. For example, an event "well W01 was shut-in in January 2009" can be described by the statements (i) Event e is of type "Well Shut-in" (e rdf:type WellShutin), (ii) the time event e occurred was January 2009 (e event:happensAtTime Jan/2009), and (iii) the domain object that event e happened to was well W01 (e event:happensTo W01). Such representation is close to the natural language approach of describing events, and also explicitly reveals the relationships (e.g. categories of events, temporal/spatial relationships) between objects.

It is also intuitive to represent domain knowledge of possible event relationships using the rule language SWRL. The format of SWRL contains a rule head and a rule body. The rule body defines a set of statements (triples) as conditions, and the rule head consists of a set of statements as consequences. The rule is of Horn format, which means if all the rule conditions are true, the conclusions are true. For example, we can include temporal conditions (time of occurrence) and spatial conditions (field entities that the event relates to) in the rule definition.

Lower coding complexity: Our event correlation framework using semantic web technologies has lower code complexity from both the system developer's point of view and the domain specific application developer's point of view. From the system developer's point of view, because the semantic web stack provides open standards of ontology languages (OWL), query languages (SPARQL) and rule languages (SWRL), we can reuse off-the-shelf tools that are developed for these semantic web standards, such as interfaces for creating and loading ontologies, rule based reasoners, etc. This reduces or eliminates the need to write any custom code for rule-based inferencing. From the domain developer's perspective of view, it is easier to code with SPARQL query language and the SWRL rule language than programming languages such as C#, Java. Moreover, it is possible to build user friendly GUI based on the well structured OWL schema and the language standards (SPARQL and SWRL).

Flexible framework: Our framework is flexible enough to capture rules of various complexity (not just "decoding keys"). Besides the classification rules defined in the WEST use case, our framework also supports regular SWRL rules, which are more descriptive. For example, we can include rich information such as the type of the well (injection or production), the time of the occurrence of the event (an instant time point or a time interval), etc., in the rule conditions.

However, semantic web technologies are relatively new (compared to, say, relational databases) and there are challenges related to performance and scalability of the triple stores, although the situation continues to improve with the availability of an increasing number of commercial products. In addition to evaluating the performance and scalability of new offerings, our future work on the well surveillance use case includes the development of more user friendly interfaces for visualization of well status and derivations, and parallelizing the inferencing process. In the larger scope of event management, we plan to capture changes of domain status and actions performed to field entities as events, and create rules that involve reservoir and well information, as well as historical event information to represent knowledge of event relationships. Based on this, we will develop an online event management system that can correlate events happening in a particular time window as new events are continuously captured and added to the system.

Acknowledgments

References

1. S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider and L. A. Stein, "OWL Web Ontology Language Reference," http://www.w3.org/TR/owl-ref/, February 2004.
2. T. Berners-Lee, J. Hendler and O. Lassila, "The Semantic Web," Scientific American, vol. 284, no. 5, pp. 34–43, 2001.
3. C. Denney et al, "Creating a Translational Ontology," Intl. Conference on Biomedical Ontology, 2009.
4. L. Feigenbaum, B. Szekely, and L. McCullough, "Cambridge Semantics: Position Paper on the Semantic Web in the Oil and Gas Industry," in W3C Workshop on Semantic Web in Oil and Gas Industry, Houston, Texas, USA, 2008.
5. J. R. Hobbs and F. Pan, "An Ontology of Time for the Semantic Web," ACM Transactions on Asian Language Information Processing (TALIP), vol. 3, no. 1, pp. 66-85, 2004.
6. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof and M. Dean, "SWRL: A Semantic Web Rule Language Combining Owl and RuleML," http://www.w3.org/Submission/SWRL/, May 2004.
7. Jena - A Semantic Web Framework, http://jena.sourceforge.net/.
8. A. Kawazoe, H. Chanlekha, M. Shigematsu, and N. Collier, "Structuring An Event Ontology for Disease Outbreak Detection," BMC Bioinformatics, 2008.
9. F. Manola, E. Miller, and B. McBride, "RDF Primer," W3C Recommendation 10, http://www.w3.org/TR/rdf-primer/, Feburary 2004.
10. Ontology, Wikipedia Entry, http://en.wikipedia.org/wiki/Ontology.
11. D. Norheim and R. Fjellheim, "Aksio - Active Knowledge Management in the Petroleum industry," in Third European Semantic Web Conference, 2006.
12. P. Schipperijn, R. Thavarajah, A. Simonato and M. Mehdizadeh, "Automated, 'By Exception' Well Surveillance: A Key to Maximizing Oil Production," SPE Digital Energy Conference & Exhibition, April 2009, Houston, Texas, USA (SPE 123145).
13. R. Soma, A. Bakshi and V. K. Prasanna, "A Semantic Framework for Integrated Asset Management in Smart Oilfields," in Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid, 2007, pp. 119-126.
14. R. Soma, and V. K. Prasanna, "Parallel Inferencing for OWL Knowledge Bases," Intl. Conference on Parallel Processing, 2008.