Design and Implementation of a Semantic Web Solution for Real-time Reservoir Management

Ram Soma\textsuperscript{2}, Amol Bakshi\textsuperscript{1}, Kanwal Gupta\textsuperscript{3},
Will Da Sie\textsuperscript{2}, Viktor Prasanna\textsuperscript{1}
\textsuperscript{1}University of Southern California, Los Angeles
\textsuperscript{2}Chevron Corp.
\textsuperscript{3}Avanade Inc.
About CiSoft

CiSoft: Center for interactive Smart Oilfield Technologies

http://cisof.usc.edu

- USC-Chevron Center of Excellence for Research and Academic Training on Interactive Smart Oilfield Technologies
- **Established**: December 2003
- **Disciplines**: Petroleum Engineering, Chemical Engineering, Material Science, Physics, Computer Science, Electrical Engineering, Industrial Engineering
- **MS Degree** in Petroleum Engineering with emphasis on Smart Oilfield Technologies (SOFT)

<table>
<thead>
<tr>
<th>RESEARCH AREAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Integrated Asset Management</td>
</tr>
<tr>
<td>• Well Productivity Improvement</td>
</tr>
<tr>
<td>• Robotics and Artificial Intelligence</td>
</tr>
<tr>
<td>• Embedded and Networked Systems</td>
</tr>
<tr>
<td>• Reservoir Management</td>
</tr>
<tr>
<td>• Data Management Tools</td>
</tr>
<tr>
<td>• Immersive Visualization</td>
</tr>
</tbody>
</table>
Outline

- Integrated Asset Management
  - Objectives
  - Role of semantic web
  - Software development methodology

- IAM Ontology
  - Ontology design
  - Change management and dirty queries

- Remarks
  - Lessons learnt
  - Areas of interest
• What is IAM?
  – A comprehensive transformation approach to integrated oilfield operations
  – A software application that can help asset team members simulate decisions before making them

• Objectives
  – Increase integration between different functions
  – Enable asset level “what if” scenarios
  – Create a knowledge base of activities and decisions
  – Reduce risk and uncertainty in decision making

• Challenges
  – Data silos are not interoperable
  – Data is semi-structured
  – Multiple organizations and classes of users
What IAM provides to users

• Efficient access to data and information
  – Reduces time spent looking for data
  – Answers complex queries across semi-structured data sets

• Consistent view of information
  – Reconciles different views of the same information
  – Creates shared “situational awareness” of the asset

• Context of information creation and usage
  – Leads to more meaningful interpretation of data
  – Acts as organizational memory for the workflow

• Non-functional: Non-disruptive, extensible, scalable, usable, etc.
The IAM “Metacatalog”

• Problem
  – Simulation models embody different realizations of uncertainty and development strategies for an asset
  – Models are created by different user groups at different times; it is difficult to maintain consistency of assumptions
  – No intuitive search functionality available to domain experts (“Show me most recently history matched model”)

• Solution: The IAM Metacatalog
  – Metadata repository at the core of the IAM application
  – Focus on answering “What does the data mean”? (vs. “How do I access the data”)
  – Key parameters and assumptions from various models are extracted and stored in the metacatalog
  – Also stores relationships between data objects and their provenance
Why Semantic Web Technologies

- Expressivity and richness of data model
- Organic growth capability for domain models/knowledge
- Inferencing and Rule Based Reasoning
- Flexibility of querying
- Ease of domain expert to understand and contribute to domain models
- Standards based (No vendor lock-in)
- Promoted by W3C
**IAM R&D Timeline**

<table>
<thead>
<tr>
<th></th>
<th>2004</th>
<th>2005</th>
<th>2006</th>
<th>2007</th>
<th>2008</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Q4</td>
<td>Q1</td>
<td>Q2</td>
<td>Q3</td>
<td>Q4</td>
</tr>
<tr>
<td>1</td>
<td>Pre semantic web work (Model based computing)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Semantic SOA for IAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Semantic web for metadata catalog (Research Prototype)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Development of IAM solution</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Semantic Web Research</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• Most of the metadata is captured offline
• Metadata extraction by custom built parsers
Example 1: Browse And Search

Search based on metadata

Provenance and metadata info for IAM data objects

Data objects
Example 2: Comparing Assumptions

Comparing a key assumption made in two simulation cases
Software Development Methodology

• Agile development using Scrum
  – Iterative software development in “Sprints”
  – Close collaboration with customer
    • Reviews/demos after each sprint
    • Flexible prioritization at sprint boundaries
    • “Product Owner” role represents the stakeholders
  – Less focus on formal documentation
Phases

- Development in sprints

• Observation: Ontology frequently modified
  - Techniques for change management make methodology more successful
Miscellaneous

• Addressed key risks of an OWL-based solution
  – Performance - Benchmarking
  – Limited tool support – Web service interfaces for KB
  – Ongoing evaluation of alternatives

• Tech transfer to software developers
  – Code and documentation
  – Demos and training

• Development
  – Ontology design was done with the assistance of domain experts and end users
  – CiSoft researcher acting as “Product Owner” for Scrum team moved research into deployment
Outline

- Integrated Asset Management
  - Objectives
  - Role of semantic web
  - Software development methodology
- IAM Ontology
  - Ontology design
  - Change management and dirty queries
- Remarks
  - Lessons learnt
  - Areas of interest
Ontology Design

- Ontology design divided into three levels to improve modularity
  - Domain independent/Upper ontologies
    - Concepts common to all ontologies like time, units etc.
  - Domain ontology
    - Model of the elements in the asset
    - Uses elements from upper ontologies
  - Application specific ontologies:
    - Elements specific to a given application or workflow
    - Uses elements from upper and domain ontologies

<table>
<thead>
<tr>
<th>Time</th>
<th>Units</th>
<th>Domain Independent/Upper Ontologies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Domain Ontology</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MDC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Events</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Application Specific Ontologies</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Domain Ontology</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDC</td>
</tr>
<tr>
<td>DSE</td>
</tr>
<tr>
<td>Events</td>
</tr>
<tr>
<td>Application Specific Ontologies</td>
</tr>
</tbody>
</table>

Tool specific Ontologies
IAM Ontologies: Domain Ontology
IAM Ontologies: Metadata ontology

- DomainElement
  - Allocation
  - LogicalEntity
  - OilfieldEntity
  - OIPEstimate
  - SimulationModel
  - Trajectory
- OOIRegionMD
- p4:Binding
- p4:DomainObjectInventory
- p4:MetadataElementMD
  - p4:IAMUserMD
  - p4:LogicalEntityMD
  - p4:RecoveryCurveMD
- p4:SimulationModelMD
  - p4:ChearsDeckMD
    - p4:ChearsHistoryFileMD
    - p4:ChearsIncludeFileMD
    - p4:GapModelMD
    - p4:MBallModelMD
    - p4:ProsperModelMD
- p4:SimulationResultMD
  - p4:ChearsSimulationResultMD
  - p4:OOIRegionSimulationResultMD
  - p4:Simulator
- p4:AccessInfoProperty
- p4:comments (multiple string)
- p4:createdOn (single dateTime)
- p4:fileName (single string)
- p4:filePath (single string)
- p4: includedInElement (multiple p4:MetadataElementMD)
- p4: includesElement (multiple p4:MetadataElementMD)
- p4: lastModifiedOn (multiple date)
- p4: objectCreatedByApplication (multiple p4: Simulator)
- p4: objectCreatedByUser (multiple p4: IAMUserMD)
- p4: objectModifiedByUser (multiple p4: IAMUserMD)
- p4: ProvenanceProperty
  - DimensionCells (multiple int)
  - DimensionCells (multiple int)
  - kDimensionCells (multiple int)
  - reservoirModelHasMaterialBalanceRegion (multiple MaterialBalanceRegion)
  - reservoirModelHasWellModel (multiple WellModel)
  - reservoirModelIsCoarseModelOfReservoirModel (multiple ReservoirModel)
  - reservoirModelIsFineModelOfReservoirModel (multiple ReservoirModel)
  - reservoirModelIsParameterized (multiple boolean)
  - maturityLevel (multiple string)
  - name (multiple string)
  - simulationModelBelongsToScenario (multiple ScenarioRoot)
  - simulationModelHasSimulationResult (multiple SimulationResult)
  - UUID (multiple int)
Implementation

• OWL data store + SPARQL querying
• Current implementation uses Jena OWL API
  – Two reasoners
    • Rule based reasoner (Jena)
    • Tableaux reasoner (Pellet)
  – OWL data stored in Jena RDBMS, file system
• Web service API to abstract data store (Apache Axis2)
• Various applications that use MDC
Supporting Iterative Development

- Ontologies are modified in every sprint
Detect *dirty queries* that are invalidated when an ontology is modified.
Dirty Queries

\[ \exists T (Q) \cap (WF'_{T,OWL} \setminus WF_{T,OWL}) \neq \emptyset \]
Change handling

- Detect ontology changes
- Evaluate Query, EXT(Q)
- Compute the impact/semantics of changes, $WF'_{T,OWL} \setminus WF_{T,OWL}$
- Match query and changes
Implementation

- Protégé plugin
  - Jena, Pellet, SPARQL parser
Outline

• Integrated Asset Management
  – Objectives
  – Role of semantic web
  – Software development methodology

• IAM Ontology
  – Ontology design
  – Change management and dirty queries

• Remarks
  – Lessons learnt
  – Areas of interest
Lessons learnt

- Ontology design
  - Plan schema changes carefully and do not change schema often
  - Keep OWL ontology small and modular; use OWL imports

- Performance
  - Track performance through product development cycle
  - Consider performance enhancing components (caching) in architecture

- Be cognizant of OWL features your tool supports
  - Very few are fully compliant with standards

- Design for change
  - Use SPARQL querying
  - Separate KB querying components from business logic and UI
  - Active area of work - expect big improvements soon
Features we missed

- **SPARQL**
  - Rollup/aggregation queries. E.g. get the aggregate of OOIP for region as sum of OOIPs of contained regions
  - Results as triples
  - XPath like expressions. E.g. get sub-tree under X

- **Updating materialized OWL knowledge bases**
  - *Solved problem* in research

- **Better XML-OWL/RDF interoperability**
  - SPARQL-XML (?)
  - OWL/RDF- XML (Gloze)
Areas of interest

• Ontology extension
  – Modeling events
  – Capturing data provenance
• Performance improvements
  – Developing representative benchmarks
  – Evaluating various RDF triple stores
  – Algorithms for parallel OWL inferencing
• Change management
  – Managing evolution of schema and instance data
  – Efficient techniques to track changes to OWL KBs
Some of our publications

- R. Soma, Viktor Prasanna, Detecting dirty queries during iterative development of OWL-based applications, 7th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2008), Monterrey, Mexico, Nov 11 - 13, 2008.
- R. Soma, A. Bakshi, V. K. Prasanna, and W. Da Sie, A Model-Based Framework for Developing and Deploying Data Aggregation Workflows, 4th International Conference on Service Oriented Computing (ICSOC), December 2006.
Backup
• Well studied problem
  – All changes to OWL, representation, capture..
• Use Protégé plugin
• Evaluate triple patterns (TP)
  – “Projecting” TP to $WF_{T,OWL}$
  – Observations:
    • All OWL statements are either type, property or identity assertions
    • Triple pattern can have variable or constant in each of its 3 places: $2^3 = 8$ types of triple patterns

• Evaluate graph pattern
  – Based on semantics of connectors
Semantics of change

- Not all changes modify WF
  - Lexical Changes: Names of entities, properties, easy to handle
  - Extensional: Modifies WF
  - Assertional: Does not change WF but adds rules
  - Cardinality: Does not change WF but adds/removes constraints

- Determine $WF'_{\text{OWL}} \setminus WF_{\text{OWL}}$ from changes
  - About 50 kinds of changes to OWL ontology

<table>
<thead>
<tr>
<th>Object</th>
<th>Operation</th>
<th>Argument(s)</th>
<th>Semantics of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ontology</td>
<td>Add_Class</td>
<td>Class definition (C)</td>
<td>IOC ≠ IOC'</td>
</tr>
<tr>
<td>Ontology</td>
<td>Remove_Class</td>
<td>Class ID (C)</td>
<td>IOC ≠ IOC', CEXT(SC) ≠ CEXT'(SC), CEXT(Dom(P)) ≠ CEXT'(Dom(P)), CEXT ≠ CEXT'(Ran(P)) ∨ P</td>
</tr>
<tr>
<td>Class (C)</td>
<td>Add_SuperClass</td>
<td>Class ID (SC)</td>
<td>CEXT(SC) ≠ CEXT'(SC)</td>
</tr>
<tr>
<td>Class(C)</td>
<td>Remove_SuperClass</td>
<td>Class ID (SC)</td>
<td>CEXT(SC) ≠ CEXT'(SC)</td>
</tr>
<tr>
<td>Property (P)</td>
<td>Set_Transitivity</td>
<td>Property ID</td>
<td>- (Assertional Change)</td>
</tr>
<tr>
<td>Property (P)</td>
<td>UnSet_Transitivity</td>
<td>Property ID</td>
<td>- (Assertional Change)</td>
</tr>
</tbody>
</table>
Matching

- Aggregate changes
- Handle Lexical change: String search/replace
- Compare extension of query with semantics of change
  - If they have some element in common ➔ dirty
  - E.g. \( \text{EXT}(Q) = P(\text{ALL}_\text{Persons} X \text{rdf:type} X \text{Person}) U P(\text{ALL}_\text{Persons} X \text{IOP} X \text{I} U \text{L}) \)
  - \( \text{Sem}(\text{ch}) = \{\text{ALL}_\text{Persons}' != \text{ALL}_\text{Persons}\} \)