Drug Discovery: An Innovative Approach to Application Integration using SOA & Ontology

Executive Summary

Service Oriented Architecture (SOA) has radically changed the application integration landscape. However, the need for intelligent service discovery based on invocation context and service capability is still a challenge. Ontology-based techniques are being experimented with as a possible answer for creating semantically enriched SOAs. In this paper, we discuss the ongoing experimentation with using ontology as a layer above Web services to achieve semantic integration. We have chosen an area of drug discovery called sample management where life sciences companies face application integration challenges. We have created a conceptual ontology model for life sciences for use in our studies. This paper first details the experiment process and then demonstrates how to achieve semantic integration using ontological means.

Introduction

The fundamental driver for any integration effort is the need to share information within the enterprise and with business partners. Increasingly it is becoming evident that enterprises that have most effective information sharing are the ones that remain competitive and successful. Due to the large heterogeneity of the IT landscape in most large enterprises, enabling application integration is not an easy task. Application software integration, as a field, has received considerable attention over the past two decades as a result of expanding business requirements (a product of functional silos built around disparate and disconnected systems) and technological advances aimed at eliminating this challenge, once and for all.

Businesses need to operate in a global environment with the ability to rapidly respond to market situations. The ecosystem of partners, suppliers, vendors need to be seamlessly integrated in a highly flexible manner. Extensible Markup Language (XML) has emerged as a powerful self-describing language to enable businesses to share information and conduct transactions on the Internet. The emergence of XML as a standard, to a large extent, has driven the evolution of application integration technologies.

Primitive efforts in application integration were focused on created specific point-to-point solutions that were usually custom built. Then, innovative middleware products came into being. These middleware products typically formed a central hub through which all integration happened. Most recent advances tap Service Oriented Architectures (SOA) to enable application integration. SOA consists of services that offer interoperability capabilities built into the network.
SOA can be considered as a business-centric approach for enabling integration. Visibility, interaction and effect are the key concepts in any SOA implementation. Visibility refers to the capacity of those with a need to see those with a capacity to service the needs. This is accomplished by publishing the service in a registry. Interaction is the act of using the capacity. Typically, the interaction is performed through message exchange. The purpose of using the capacity is to get a real world effect. This effect may be a changed data state or a file transferred, for example.

Technically, SOA is most often realized by using Web services. Web services follow the “Find, Bind, Invoke” paradigm wherein the service that needs to be used is first found and then bound and invoked. The second and the third steps of “binding and invoking” are pure technical problems -- usually solved by using Web Service Definition Language (WSDL) and Simple Object Access Protocol (SOAP) messages. It is the first part of “finding” the service (which can be theoretically solved by using Universal Description, Discovery, and Integration, better known as UDDI), that presents the most interesting problem -- how do we find the right service to be used?

The real world demands that systems have the ability to intelligently make appropriate selection of services. An inference-driven platform for SOA can provide the ability to make runtime decisions based on service capabilities and the context of invocation. SOA by itself does not have this capability. It is in this context that semantic richness can be of tremendous help. Semantic richness can be thought of “as a way of modeling a domain as close to natural language as possible”. The primary goal of this paper is to demonstrate how semantic technology can be used to create an inference-driven platform for SOA.

The rest of this paper is organized as follows. The next section provides the background of semantic integration and ontology. After that, the paper discusses our experimental study of semantic technology and outlines the ontology model. We conclude by offering a direction for future research.
Semantic Integration and Ontology

Semantic integration techniques attempt to build context or awareness about the underlying services or data in the data sources in a meta-layer. Semantic Integration techniques are based on the axiom: Heterogeneous data (for a domain) may, on analysis, reveal patterns of similarity in their origin, making it possible to define these patterns via a meta-model and to inter-relate the meta-models by defining relationships.

Ontology-based approaches are considered as one way of achieving semantic integration.

Before we proceed to ontology-based SOA, a brief background of ontology is relevant. Any area of knowledge (domain) is usually expressed by splitting the domain to "classes" and by expressing relationships between those "classes". Ontology is defined as a “structured representation of a domain in terms of classes and interrelationships between the classes.” For example, if we want to understand project management, the ontological approach would be:

Fig.2 Ontology chart for project management (from Lui, 2000, pp. 79)

Project Management (Domain)

- Person (Class)
  - Department (Individual)
- Organization (Class)
  - Department (Individual)
  - Project (Individual)
  - Employer (Individual)

There are several relationships possible between classes: person lives in a society, organization exists in a society, employee has an employer, project has tasks, organization defines a project, employee works on a project, employee is assigned to a task, etc.

Ontology enabled SOA\(^8\) tries to extend and use this modeling technique. Computer science requires ontology to be specified in the form of Web Ontology Language (OWL) for processing. OWL is an XML-based specification. Fundamentally, ontology-enabled SOA resembles the following:
Briefly, in an ontology-enabled SOA, a layer is created that resides on top of the services dictionary that contains the ontology mappings of the services domain. This is mapped with sufficient data during the design time which is then used during the runtime to find semantic similarity of the service request. Policies are built that contain logic for finding semantic similarity.

Selene: An Experimental Study

Selene is a Cognizant-designed architectural framework created to understand the efficacy of using ontology to enable semantic integration. The rest of this section focuses on the description of the problem domain that was addressed, the architectural solution and study results.

The Problem Domain

Drug discovery is an intensely collaborative exercise wherein several scientists working in various disciplines and systems are involved. It is a highly data-driven process where huge volumes of data are created and analyzed that needs to be shared with numerous scientists. The more the number of people and systems involved in any activity, the greater the information integration challenges. This is especially pronounced in an intensely R&D-based activity like drug discovery where various types of applications are used by molecular biologists, biochemists, toxicologists, pharmacologists, etc. The existing integration techniques that focus on data and their drawbacks are dealt with more extensively in whereas the authors also lay a strong case for the need for semantic techniques.

In the recent past, business realities have compelled drug discovery to embrace a collaborative approach involving several partners. This has opened up additional application integration challenges, particularly as the IT landscape has diversified. Initiatives like SIMDAT to enable collaboration in the drug discovery realm have gained momentum. As a result, many R&D organizations in the pharmaceuticals business now use SOA as an effective medium for integration across the islands of information silos. Consider, for example, an application that needs to be built integrating various data sources like:

- Chemistry
- Biology
- Materials
- Lab
- Assay Results
- Syndicated Data (Public Domain and Subscribed)
- Scientific Journals

One can create various atomic services from these underlying data sources using a technology platform that enables the composition of applications by linking services together.

Selene is a Cognizant-designed architectural framework created to understand the efficacy of using ontology to enable semantic integration.
This paper focuses on a particular problem domain within drug discovery area called sample management. All drug discovery activity needs chemical and biological samples. Sample management deals with the way the samples are acquired, registered, stored and distributed. It can be defined as “activities in and around sample acquisition, registration and supply of samples for drug discovery purposes.” Sample management thus involves dealing with a variety of data sources for registration, storage and distribution of samples.

The figure below depicts typical areas that fall under sample management.

For the purpose of Selene, we chose an area of sample management called sample acquisition. Sample acquisition deals with the act of acquiring samples that are required for assay purposes. The samples can be ordered by scientists. The following table illustrates the “points of variability” where semantic integration is required. Points of variability are the areas where inference capability is required to have semantic richness.

<table>
<thead>
<tr>
<th>Variability Matrix</th>
<th>Request</th>
<th>Vendor Selection</th>
<th>Ordering</th>
<th>Identification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquisition Process</td>
<td>Raised by [Scientist, Branch, HTS, Ext. Research Org.]</td>
<td>Source [Chemists, Pharmaceutical Company, Commercial Sample Library]</td>
<td>Vendor [For (Vendor Selection source)]</td>
<td>Type [Chemical, Biological]</td>
</tr>
<tr>
<td>Type [Chemical, Biological]</td>
<td>Container Type [Vial, Plate]</td>
<td>Sample State [Solid, Liquid]</td>
<td>Container Type [Vial, Plate]</td>
<td>Uniqueness Logic [SMILES, Sequence]</td>
</tr>
<tr>
<td>Name [Common, IUPAC]</td>
<td>Molecular Formula</td>
<td>Purity/Concentration</td>
<td>Location/Site</td>
<td>Bar Code</td>
</tr>
<tr>
<td>Sample State [Solid, Liquid]</td>
<td>Container Type [Vial, Plate]</td>
<td>Purity/Concentration</td>
<td>Location/Site</td>
<td>Bar Code</td>
</tr>
<tr>
<td>Container Type [Vial, Plate]</td>
<td>Quantity [Volume, Weight]</td>
<td>Purity/Concentration</td>
<td>Location/Site</td>
<td>Bar Code</td>
</tr>
<tr>
<td>Quantity [Volume, Weight]</td>
<td>Purity/Concentration</td>
<td>Purity/Concentration</td>
<td>Location/Site</td>
<td>Bar Code</td>
</tr>
<tr>
<td>Purity/Concentration</td>
<td></td>
<td>Purity/Concentration</td>
<td>Location/Site</td>
<td>Bar Code</td>
</tr>
</tbody>
</table>
As can be seen from the table, there are several variables for each step. It may be that these variables are today well defined and handled appropriately by existing systems. However, future business needs may mandate introduction of additional variables or simply new values for existing variables (e.g., a new container type or ability to handle gaseous state samples). Ideally we would like to have a service definition, for example “Sample Registration service,” which should not require changes for incorporating the new variables. Similarly, the need to incorporate a new vendor to supply samples, for example, should not mandate changes to existing acquisition service implementation. Here is where defining an ontology model for these variables enables an extensible vocabulary and a “non-invasive” way to meet new business needs.

Solution Architecture

Our architectural solution, christened “Selene” was created as an SOA-based framework. We used IBM products to develop this architecture. It contains workflow tasks and core business processes designed as Service Component Architecture (SCA) components with Java implementations.

The architecture consists of the following broad components:
- Business Process Modeler
- The Ontology Model
- Context Sensitive Invoker
- Web Services

Business Process Modeler

Business Process Modeler is used to model the business process into discrete tasks. The “Sample Acquisition” business flow is modeled as follows:
Briefly, the workflow of sample acquisition is as follows. A scientist requests a sample. This sample may either exist in the inventory, may have been in inventory at some time but is now depleted, or may be a completely new sample (i.e., never had an entry in the inventory). The pharmaceutical company may have a list of preferred vendors for supplying samples; the scientist can choose to place the order with one of these vendors. After the sample is ordered, when it is received, depending on whether the sample existed earlier in inventory or not, registration of the sample is required. Once the sample is registered, it is available for dispersal.

This flow includes three main human tasks:

- Raise Sample Request
- Choose Preferred Vendor to Order
- Receive Sample

Once the process is defined, it is expressed in Business Process Execution Language (BPEL) form in the Business Modeler.

The Ontology Model

As described in the introduction, Ontology is defined in terms “concepts” and “relationships.” Concepts are implemented in ‘classes’ while “relationships” are implemented in terms of “Object Properties” and “Data type properties”.

For the sample acquisition process that is described in the previous section, we define a conceptual ontology model. The following are the main classes in our model:

- Sample
- Sample Request
- Sample Order
- Scientist
- Lab Staff
- Vendor

For the purpose of simplicity, in our ontology we have chosen to represent the below subset of the classes and their relationships, based on the aforementioned variability matrix. These classes are grouped under appropriately named super classes as shown in the following table.
The **Actor** class is a suitable abstraction of entities like Vendor and Scientist. Similarly a Sample Request or Sample Order is a kind of Document while **Material** appropriately abstracts an entity like Sample.

Similarly we use Attribute class to represent attributes and their possible values (for example, “Sample Type” is an Attribute of the Material, “Sample,” can have a value of “Chemical” or “Biological.”)

The above definitions and classifications allow for the visualization of the classes and their relationships in a generic manner conforming to ontology representations as shown below:

<table>
<thead>
<tr>
<th>Super Class</th>
<th>Class</th>
<th>Relationship</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Material</td>
<td>Sample</td>
<td>hasName</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hasSampleType</td>
<td>Sample Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hasState</td>
<td>Sample State</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hasContainerType</td>
<td>Container Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hasStorageTemperature</td>
<td>Storage Temperature Structure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hasStructure</td>
<td></td>
</tr>
<tr>
<td>Document</td>
<td>Sample Request</td>
<td>contains</td>
<td>Sample Requested Quantity Requester (Lab staff)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hasRequestedQuantity</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>hasRequester</td>
<td></td>
</tr>
<tr>
<td>Document</td>
<td>Sample Order</td>
<td>contains</td>
<td>Vendor Sample Request</td>
</tr>
<tr>
<td></td>
<td></td>
<td>contains</td>
<td></td>
</tr>
<tr>
<td>Actor</td>
<td>Scientist</td>
<td>isEmployedWith</td>
<td>Organization</td>
</tr>
<tr>
<td>Actor</td>
<td>Lab Staff</td>
<td>isEmployedWith reports to</td>
<td>Organization Scientist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>reports to</td>
<td></td>
</tr>
<tr>
<td>Actor</td>
<td>Vendor</td>
<td>hasName</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hasRating</td>
<td>Rating</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hasVendorType</td>
<td>VendorType</td>
</tr>
</tbody>
</table>

Or more specifically:

![Diagram showing relationships between classes and attributes](image)
For our ontology, we could represent the relationship between the Actor (Vendor) and the Attribute (Name) with the Object Property “hasName.” The same property can also depict the relationship between the Material (Sample) and Attribute (Name). In other words, we can say that the domain for the object property “hasName,” includes the Actor (Vendor), and the Material (Sample), while the Attribute (Name) is in its range. This can be expressed in OWL as below:

```xml
<owl:ObjectProperty rdf:ID="hasName">
  <rdfs:range rdf:resource="#Name"/>
  <rdfs:domain rdf:resource="#Vendor"/>
  <rdfs:domain rdf:resource="#Sample"/>
  <rdfs:domain rdf:resource="#SampleState"/>
  <rdfs:domain rdf:resource="#SampleType"/>
  <rdfs:domain rdf:resource="#VendorType"/>
  <rdfs:domain rdf:resource="#Structure"/>
  <rdfs:range rdf:resource="#SampleOrder"/>
  <rdfs:range rdf:resource="#Document"/>
</owl:ObjectProperty>
```

The diagram below helps to visualize our ontology model for the sample acquisition process.

The runtime ontology is created using Rational Software Architect 7.5. Currently, Selene uses “Vendor Name” and “Sample Type” assertions to define end point selection policies. Other
assertions are included for possible future use. The runtime ontology mapping as represented in the Unified Modeling Language (UML) form is shown below:

Selene uses IBM Fabric 6.2 as the core ontology engine. The ontology, as implemented in Selene, achieves portability and scalability through loose coupling. As an example, a pharmaceutical company can forge partnership with a new supplier for samples. Selene would need no change to its core business process as implemented in BPEL. Extending the ontology for the new vendor and defining the new end point with a policy assertion binding would suffice. Of course, the new vendor would need to expose its business interface as a Web service either registered on a UDDI registry or share a library of services expressed in WSDL.

Context Sensitive Invoker

The business flow is encapsulated as a linear process in a BPEL block. The “Context Sensitive Invoker” uses ontology mapping to determine the appropriate Web service. This is done by “Assertions” and “Policies”. Policies operate on the available and published end points. Assertions are the ontology mappings of all possible context-sensitive variables. Policies are defined in terms of these assertions -- usually a subset of assertions. Context Sensitive Invoker determines at runtime the end point to invoke based on policy assertions.

Web Services

Web services provide the actual business service. For Selene these are considered as “consuming” points in the sense that Web services are invoked. From an architectural framework perspective, the operational assumption is that Web services will be made available for use.

The policy to be defined on the Web services is very much part of the tasks required to incorporate a new service endpoint.

Study Results

The diagram on the following page illustrates how the Selene architecture works at runtime.

A BPEL process is initiated on receipt of a “Sample Request” which contains sample criteria like sample state, preferred vendor, etc.
The BPEL engine of the “Business Process Modeler” passes this on to the “Context Sensitive Invoker.” The SCA components of Context Sensitive Invoker interact with the “Ontology Engine” and retrieve the assertions based on the context which in this illustrated case is “Vendor.”

Thus the desired context-sensitive selection of Web services is gained using ontological means. We further note that it is relatively simple to achieve this using ontology as the medium for semantic integration.

Conclusions & Future Work

In this work, we have shown that it is possible to use ontology for enabling semantic integration. The approach followed was to model the business domain process as a workflow. Then the points of variability were identified for each of the workflow nodes and the decision points. The ontology model construction was a parallel activity that happened alongside the business process modeling. The classes and relationships of the ontology model are created based on the domain knowledge and the business process model. This ontology was then converted to Web ontology language form for computational purposes. The context-sensitive invoker, which is the core of this study, determines the runtime invocations based on the ontology model. Thus, we have achieved semantic integration using ontology.

Further work needs to be done to create a more complete ontology that encompasses not just sample acquisition activity but the entire sample management operations.
The ontology that is shown in this paper is not complete. Further work needs to be done to create a more complete ontology that encompasses not just sample acquisition activity but the entire sample management operations. We plan to strengthen the domain ontology model by adding more domain specific information. Of relevance, is the SOA-based ontology model proposed by The Open Group. We intend to create an ontology model based on the approach defined by The Open Group.

From a technical perspective, we feel that ontology is a powerful approach for creating intelligent software agents. Work on using ontology for query reformulation represents an interesting possibility for use of ontology. Such perceptive ability is also required in software agents to make runtime decisions; we believe that it can be built using ontology. This can help in creating user friendly software that helps in decision making. For example, in the domain of drug discovery, we can create a software agent that using ontology analyzes the sample requests of the scientists and suggests potential alternatives across the vendor catalogues. We plan to investigate this approach.

Acknowledgements

The authors thank IBM for their support in realizing the Selene architecture. We thank Rakesh Ronald, Business Analyst at Cognizant, for his timely and valuable help in modeling the business process. We express our gratitude to Balasubramanian Sankaranarayanan, Practice Director for Life Sciences at Cognizant, for his encouragement and guidance in this effort.

About the Authors

Raghuraman Krishnamurthy is a Principal Architect in Cognizant’s Technology Consulting Group, working within the life sciences domain. His focus spans software development and linking IT to business and software architectures. Raghu’s technical area of expertise includes enterprise architecture, SOA and application integration. He has been involved in various consulting engagements in architecture assessment, application integration. Raghu holds a Master’s degree from the Indian Institute of Technology, Mumbai. He is a TOGAF certified enterprise architect.

Vinod Ranganathan works as a Technical Lead on J2EE/SOA platform projects for various pharmaceutical clients of Cognizant. He has considerable experience in the areas of application development, project management and technical consulting. Vinod is also involved in design and execution of PoCs in addition to architecture recommendations and software estimation for potential projects. He holds a Masters degree in Biotechnology from Pune University and a Diploma in Advanced Computing from C-DAC. He is also working towards a Post-Graduate Diploma in Business Administration from Symbiosis Institute.

Baskar Senguttuvan is a Senior Architect in Cognizant’s Technology Consulting Group working within the life sciences domain. He has expertise in architecture consulting and project management. Baskar has consulted with various pharmaceutical clients in the areas of SOA and information technology architecture, among other things. He holds Bachelor’s Degree in Computer Science and MBA in Systems. He is certified PMP and SEI Software Architecture Professional from Carnegie Mellon’s Software Engineering Institute.
References


3. Wikepedia on Semantic Integration, Ontology and Service Oriented Architectures


6. IBM literature on SOA and tools


1. SIMDAT http://www.simdat.eu/859.html


About Cognizant

Cognizant (NASDAQ: CTSH) is a leading provider of information technology, consulting and business process outsourcing services. Cognizant’s single-minded passion is to dedicate our global technology and innovation know-how, our industry expertise and worldwide resources to working together with clients to make their businesses stronger. With over 50 global delivery centers and 63,700 employees as of March 31, 2009, we combine a unique onsite/offshore delivery model infused by a distinct culture of customer satisfaction. A member of the NASDAQ-100 Index and S&P 500 Index, Cognizant is a Forbes Global 2000 company and a member of the Fortune 1000 and is ranked among the top information technology companies in BusinessWeek’s Hot Growth and Top 50 Performers listings.

Start Today

For more information on how to drive your business results with Cognizant, contact us at inquiry@cognizant.com or visit our website at www.cognizant.com.