RC24186 (W0702-070) February 16, 2007
Computer Science

IBM Research Report

Ontologies for Model-Driven Business Transformation

Juhnyoung Lee
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Ontologies for Model-Driven Business Transformation

Juhnyoung Lee

IBM T. J. Watson Research Center
Hawthorne, New York 10532
jyl@us.ibm.com

Abstract

Semantic markup languages such as RDF (Resource Description Framework) and OWL (Web Ontology Language) are increasingly used to externalize metadata or ontology about business data, software and services in a declarative form. Such externalized descriptions in ontological format are utilized for purposes ranging from search and retrieval to information integration and to business transformation. Ontology can significantly reduce the costs and improve the qualities of deploying, querying, integrating, and transforming enterprise systems. This paper presents an innovative application of ontology to a model-driven approach to business analysis and transformation. The approach employs a daisy chain of business models for causality analyses. It links, by using semantic models, business processes and business components to IT solutions and capabilities at different phases of business transformation. The semantic models help infer causality of any business pain points and recommend appropriate solutions to fix business or IT shortfalls associated with the pain points in the process of business transformation. In addition, this paper presents an enterprise-scale ontology management system which provides functionality, scalability and performance demanded by enterprise applications such as the proposed model-driven business transformation. It describes the design and implementation of the management system which programmatically supports the ontology needs of business applications in a similar way a database management system supports their data needs.

Keywords: Semantic Web, ontology, ontology management system, inference, business transformation, model-driven architecture

1. Introduction

Ontology is similar to a dictionary, taxonomy or glossary, but with structure and formalism that enables computers to process its content. It consists of a set of concepts, axioms, and relations, and represents an area of knowledge. Unlike taxonomy or glossary, ontology allows to model arbitrary relations among concepts, also model logical properties and semantics of the relations such as symmetricity, transitivity and inverse, and logically reason about the relations. Ontology is specified in a declarative form by using semantic markup languages such as RDF [18] and OWL [21]. It provides a number of potential benefits in processing knowledge, including the separation of domain knowledge from operational knowledge, sharing of common understanding of subjects among human and also among computer programs, and the reuse of domain knowledge. In general, ontology can be beneficial to any enterprise system dealing with multiple domain concepts that are interrelated and needs to use the concepts to describe the behavior or capabilities of its programs. Business

application examples of ontology include business process integration by using Web services composition, information retrieval and search systems for semantic-based search capabilities, video retrieval systems to annotate media with metadata, and business collaboration management using corporate social network to provide a common understanding to collaboration contexts and annotate them, to name a few.

Among the enterprise applications of ontology, this paper focuses on its use in business transformation processes. Business transformation is a key executive management initiative that attempts to align the technology initiatives of a company closely with its business strategy and vision, and is achieved through efforts from both the business and IT sides of the company. However, the technology side of the company often emphasizes functions and capabilities, while the business side focuses on business impact and value. Because of this "business-IT gap" [11], business transformation processes for IT and services are lengthy and costly. To address this problem, this paper presents an innovative application of ontology to a model-driven approach to business analysis and transformation. This approach innovatively extends the *Model-Driven Business Transformation* [6, 8] and utilizes semantic models that links business performance measures, business processes and components with key IT enablers all the way down to the IT infrastructure. The ontological models is used to infer both direct and indirect causalities of any business pain points, and recommend appropriate solutions to fix the business or IT shortfalls associated with the business pain points.

In the second part of this chapter, we present an enterprise-scale ontology management system which provides functionality, scalability and performance that enterprise applications such as the proposed model-driven business transformation would demand. In recent years, there has been a surge of interest in using ontological information for communicating knowledge among software systems. As a result, an increasing range of software systems need to engage in a variety of ontology management tasks, including the creation, storage, search, query, reuse, maintenance, and integration of ontological information. Recently, there have been efforts to externalize such ontology management burden from individual software systems and put them together in middleware known as an ontology management system. An ontology management system provides a mechanism to deal with ontological information at an appropriate level of abstraction. By using programming interfaces and query languages the ontology management system provides, application programs can manipulate and query ontologies without the need to know their details or to re-implement the semantics of standard ontology languages. Such a setting is analogous to the way a database management system allows applications to deal with data as tables and provides a query engine that can understand and optimize SQL queries. This paper describes the design and implementation of the SnoBase ontology management system [10], which was developed at IBM T. J. Watson Research Center.

The rest of this paper is structured as follows: In Section 2, we introduce the model-driven business transformation and briefly describe how ontologies and semantic technologies can facilitate the business transformation process. Section 3 presents a semantic model for the proposed business transformation approach. In Section 4, we explain several qualitative business analyses helping business transformation. In Section 5, we provide a schematic overview of the SnoBase ontology management system. In Section 6, we describe the design of JOBC API with examples. Section 7 presents a model-driven approach to building an

environment for the development and transformation of semantic applications and models. In Section 8, conclusions are drawn and future work is outlined.

2. Model-Driven Business Transformation

Among the emerging methods and the supporting technology for business transformation in the service-led economy is the *model-driven business transformation*. Briefly, the model-driven approach requires a model representation of business entities such as business processes, components, competencies, activities, resources, metrics, key performance indicators (KPI), and their relationships. Then, the model is utilized to identify opportunities for saving costs or improve business processes. Semantic models using ontology markup languages provide useful representation of business models because they are not limited by types of relationships among business entities. Also, the automatic reasoning capability of semantic models provides an effective method for analyzing business models for identifying cost-saving or process improvement opportunities. For example, business performance metrics naturally fit well with business activities and traditionally represented that way. By using this relationship between business activities and metrics, and also the relationship between business components and business activities represented in a semantic model, a business analyst can infer relationship between business components and metrics. This relationship can provides business insights into how the corporate can improve its performance metrics by addressing issues with the business components associated with the selected set of metrics. Then, by identifying, again in the semantic model, IT systems associated with the business components, the business analysts are able to suggest recommendations about IT system management to improve performance metrics.

More formally, the model-driven business transformation utilizes a multi-layer model approach to link business and IT semantics [6, 8]. The upper layers of the model represent business semantics in the terms familiar to business executives, business managers and analysts such as key performance indicators, operational metrics, business processes, activities and governance. The lower layers of the model represent IT architecture comprising a wide range of services implemented in IT infrastructure such as service-oriented architecture. The vision of this multi-layer model is to enable IT solutions to accurately reflect and be driven by business intent. Figure 1 illustrates the multi-layer model approach to business transformation.

The key to this multi-layer model is that the layers are linked in meaningful ways, so changes in one layer can ripple through other layers. The representation and enforcement of the semantics of the different layers and also of the connections between the layers is essential to the model-driven approach and also is an application area of the semantic Web technology. This model-driven approach provides a convergence of the business and IT models using a multi-layer model, which tightly couples the business and IT models. In many ways, this vision is not new. Technologists have been working towards generalized business process integration and automation for many years. However, this approach is different from the typical technology-oriented business integration, because it provides a top-down business perspective which enforces a business-orientation of business transformation.

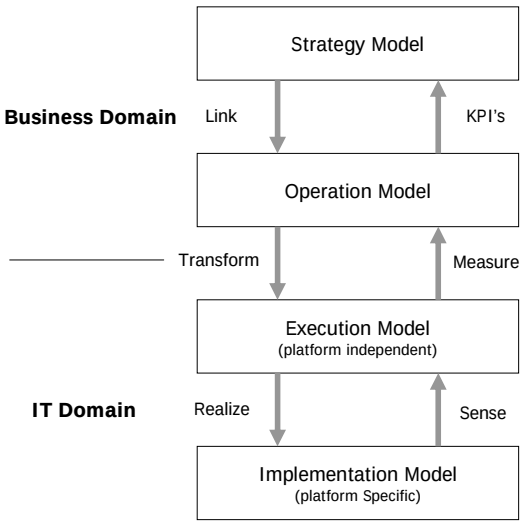

Figure 1 Model-driven business transformation

Once equipped with end-to-end tools for the model design, connection and transformation, this approach has the potential to reduce the time-to-value of business solution implementations. It would replace the manual creation of unstructured business documents and informal business models with a guided transformation of a structured multi-layer model. The IT solutions generated by this approach would accurately and precisely reflect the original business semantics and are directly deployable and executable in a service-oriented architecture. This model-driven business transformation approach is a significant step towards closing the infamous "business-IT gap" [11], achieving maintainable alignment between business design and IT solutions.

Recent trends in business and software componentization and modeling would boost this model-driven approach as a prominent methodology for the service-led economy. In recent years, enterprises componentize into discrete services to achieve operational efficiency, flexibility, and to sharpen their focus. Also, the consulting industry increasingly utilizes sophisticated modeling techniques to understand and transform businesses. In the IT domain, software modeling technologies and methodologies such as the Object Management Group's Universal Modeling Language [13] and Model-Driven Architecture [12] are widely adopted and studied in both industry and academia. In addition, W3C's Web services [23] and related technologies accelerate the shift towards service-oriented architectures [22] which fit the model-driven business transformation approach. The trends in business and software componentization and modeling effectively converge to provide new layers of business understanding and responsiveness.

Traditionally, a model has been used to mean a physical representation of some thing in various contexts including studies of physics, mathematics, statistics, economics, geology, psychology, computer science, to name a few. As we observe in examples such as the particle physics history [1], a good model capacitate the progress of the study, while a poor one limits it. A model often dominates the understanding and solution to the given problem

in the domain. Additionally, the language used to specify a model often impacts on (either assists or limits) the thinking process with the model. The most important component of the model-driven business transformation approach is the model, i.e., the representation of the semantics of business and IT resources. With the multiple layers in the model, another key component is the representation of the meaning of the links across different layers. It is crucial to this model-driven approach how we represent in a language and enforce the semantics of the layers and also of their links.

W3C's Semantic Web [19], which intends to create a universal medium for information exchange by giving semantics, in a manner understandable by machines, to the content of resources, provides an appropriate option to address this modeling requirement of the model-driven approach. The Semantic Web is comprised of the standards and tools of markup languages including XML [16], XML Schema [24], RDF [18], RDF Schema [17] and OWL [21]. These semantic markup languages would be used to specify ontological representation of models including the business and IT models and their connections. The semantic markup languages would be used to specify the convergence of business and IT models, and more importantly, their metamodels. The ontological representation of the metamodel of a constituent model enables reasoning about the instance model, which enables a dependency analysis to deduce unknown or implied relationships among entities within the instance model. The analysis would be extended across multiple layers of models. The semantic model-based dependency analysis would reveal which entity has an impact on which entities of the multiple layers of the model such as business components, business processes, key performance indicators, IT systems, software classes and objects, etc. This semantic model-based analysis can be applied to a model that provides an introspective view of the business within an enterprise. Also, it can be applied to a value network which yields an extrospective view of businesses in an ecosystem.

In addition to its use in the model-driven business transformation, the semantic model approach is also useful in business information and process integration. Suppose a business solution requires integrating a number of data sources (or application interfaces for process integration) which provide different but overlapping conceptual models. For information integration, we would start building a global conceptual model which is essentially a semantic model. Then, the data sources are defined as views into this global model, although there is no guarantee of completeness. A query to the data sources would be expressed in the global semantic model. The result set for the query would be constructed by finding all conjunctive queries over the views that are contained in the top-layer query. A semantic model-based approach to process integration would require a similar set of steps over a set of overlapping application interfaces.

The model-driven business transformation approach proposes new business methods and the supporting technology by introducing a multi-layer model which couples business and IT models. It provides a top-down business perspective which enforces a business-orientated business transformation. It has the potential to provide a number of benefits over the traditional technology-oriented approach, including business-IT alignment, reason about business design and transformation, real-time visibility into business operation, improved business performance management, rapid and repeatable IT solution implementation, and adaptive IT solution implementation. The key to this model-driven approach is that the layers are linked in meaningful ways, and that the semantics of the links are effectively

represented and reasoned. Therefore, changes in one layer can accurately ripple through other layers. The semantic Web technology provides an appropriate option for this modeling requirement by enabling representation and enforcement of the semantics of the layers of the model. It poses a key enabling technology for the emerging service science, which will meld technology with an understanding of business processes and organization.

3. Semantic Model for Business Transformation

Figure 2 illustrates part of the high-level semantic model designed for the model-driven business transformation. The model was used in a research prototype system for business transformation referred to as *VIOLA* developed at IBM T. J. Watson Research Center [9]. The model captures business entities (and their relationships) of an enterprise that are involved in creating or defining value. The business entities in the model include business components, business processes and activities, operational metrics, key performance indicators, and value drivers. In addition, the model represents their relationships to resources, services, messages, IT infrastructure, and solutions. Often, solutions refer to both IT and business capabilities to support certain business objectives and strategies, or address business pain points.

It is important to note that each business entity in this model, such as business components, business processes, operational metrics, key performance indicators, value drivers and solutions, has its own (hierarchical or networked) structure with its own constituent elements. Therefore, each forms a model in its own right. In that sense, our semantic model is a metamodel and we often call it a *daisy-chain of models*. It is the daisy-chain part that enables the causality analysis across business entities by discovering direct and indirect relationships among them through inference.

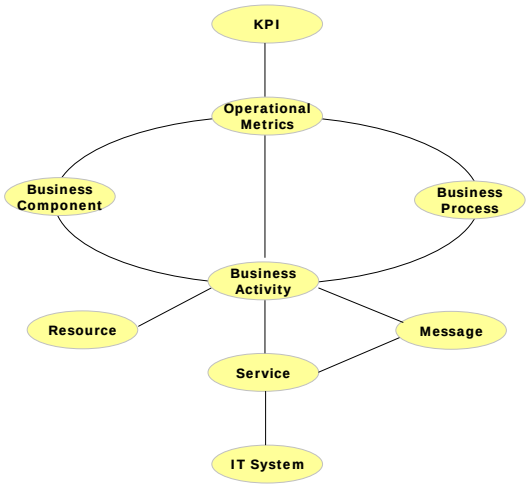

Figure 2: Semantic model for business transformation

To allow the user to explore this rich information captured by this model, we define multiple views into the model referred to as *business maps*. Each business map shows various entities involved in running and understanding of business and their relationships. Our business

maps provide visual models which organize the above-mentioned business entities in a structured way. In addition, they provide user interfaces which allows to the user interactively navigate and explore the information space for an analysis purpose.

Figure 3 shows a screenshot of a business map in the VIOLA system that contains the component business map, the value driver tree and the business activity tree. To build such a business map, we utilize industry standard taxonomies of business processes and metrics such as APQC Process Classification Framework [2], and their relationship to value drivers and business components. Additionally, we allow the user to customize the industry standards to the needs of a specific enterprise, and import and export the enterprise-specific value driver trees and component business maps.

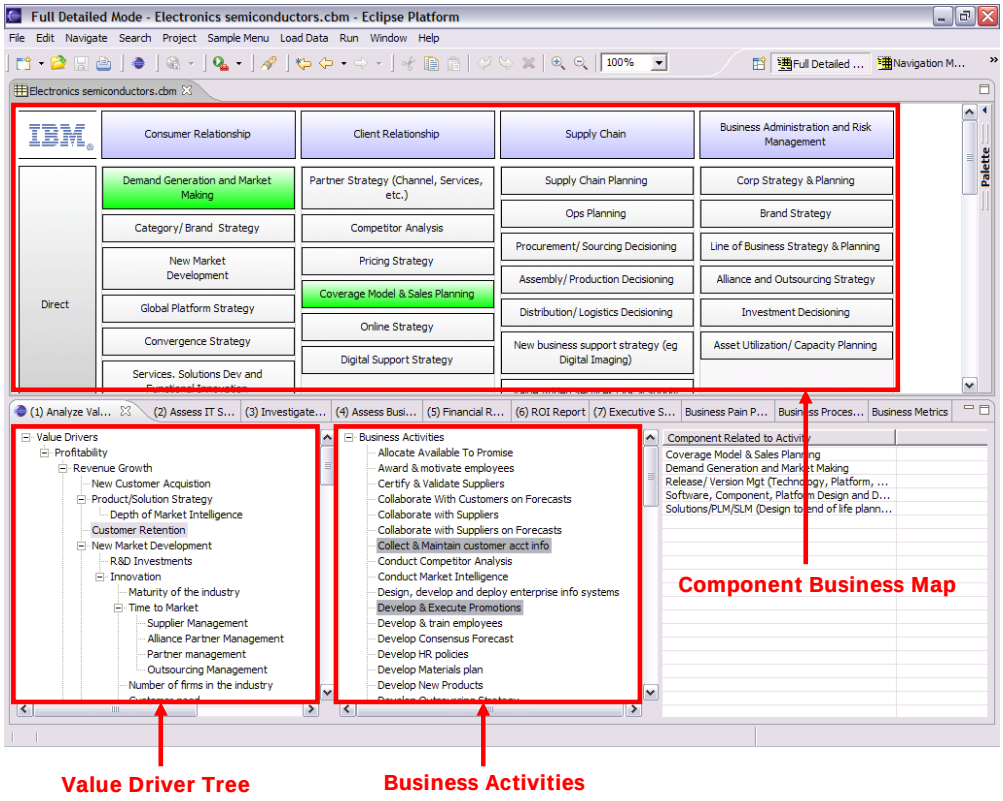

Figure 3: Business map views in VIOLA

4. Semantic Analyses for Business Transformation

The main advantage of the VIOLA semantic model is the enablement of various types of analyses that would allow the user to obtain interesting insights into the current state of a business and its possible business transformation opportunities. The VIOLA system is designed to provide the qualitative business analysis, among others, based on component business models, business process models, value driver models and solution models. By linking them together, VIOLA provides an end-to-end suite of business analysis capabilities, enabling business and value-oriented business transformation. This section describes a few qualitative analysis capabilities of VIOLA.

4.1. Dependency Analysis

The dependency analysis allows the user to explore the business maps and understand the correlations and (direct and indirect) dependencies among business entities. For examples, this capability can interactively identify one or more business components associated with a particular value driver. Conversely, it can find one or more value drivers that are affected by the performance of a particular business component. The associations between value drivers and business components are discovered through their relationships with business processes and activities. Similarly, VIOLA can identify and show dependencies between business activities and IT applications, and also between business activities and solutions, both IT and business-driven. Furthermore, the relationships are transitive, and so it is possible to infer the associations between value drivers and IT applications/solutions, also between components and IT applications/solutions, and so on.

To support the dependency analysis, the VIOLA system captures the basic relationship information in the VIOLA business model. Once the explicit relationship data are populated in the database using the model, the system utilizes a Semantic Query Engine to infer implicit relationships among various business entities by using the explicit relationships and their logical properties. To provide the inference capability, VIOLA utilized W3C's OWL semantic Web markup language [21] and the SnoBase Ontology Management system [10] developed at IBM T. J. Watson Research Center which will be described in the following sections. While OWL and SnoBase for VIOLA was a choice of convenience based on their immediate availability for the development, the semantic models used in VIOLA could be represented in knowledge model languages such as *topic map* [15], which is an ISO standard for the representation and interchange of knowledge, with an emphasis on the findability of information.

4.2. Heat Map Analysis

This analysis is an essential capability of Component Business Modeling (CBM) [7] where the user discovers one or more "hot" components that are associated with one or more business strategies and/or pain points. In the traditional CBM analysis, this step was conducted manually by the analyst depending on his/her knowledge and expertise in the business domain. VIOLA automated the capability by taking performance values into account with the dependency analysis.

First, the system allows the user to explore the value driver tree to identify one or more value drivers that may be associated with a certain business strategy/pain point. The discovery of "hot" components that affect the business strategy can be accomplished by executing a simple semantic query to the business model represented in OWL. Then the system colors the identified hot components differently to distinguish ones that affect positively or negatively to the strategy. The VIOLA system compares the industry benchmark and the as-is value of the operational metrics and performance indicators associated with the components to decide on their color. Figure 3 displays a heat map showing a couple of hot components affecting positively (as denoted by the green color) to a value driver, "Customer Retention," which is highlighted in the value driver tree.

4.3. Shortfall Assessment

The Shortfall Assessment allows the user to map the existing IT infrastructure against the "hot" components identified in the heat map analysis. It helps understand how the current IT infrastructure, such as applications and network capabilities, supports the business, especially, for those hot components. The analysis requires collecting the information on the current IT infrastructure and representing it in a semantic business model in OWL. Then the mapping of IT applications and capabilities to the components becomes, again, an execution of a simple semantic query to the semantic model.

VIOLA visualizes the mapping on the CBM map by overlaying IT applications on components. Then, the user can visually identify possible IT shortfalls and classify them into several types. Typically, four types of opportunities tend to arise. First, a *gap* indicates that a hot component does not have any IT support. The enterprise may want to consider an IT investment to improve the component's performance and support the intended business transformation. Second, a *duplication* indicates that a component is supported by multiple IT applications, possibly, deployed over time. The business may want to consolidate the applications to improve performance and reduce cost in communication and maintenance overhead. Third, a *deficiency* indicates that the current application lacks key functionality, or is poorly designed, and so incurs a project opportunity. Finally, an *over-extension* indicates that a system designed to support one business component is extended beyond its core capability to support others. Different definitions for the shortfall types may apply. With precise definitions of the shortfall types, the VIOLA system also automates the shortfall classification and recommends to the user the initially identified shortfalls.

It is important to note that an IT system can be involved with multiple situations. The value model of the VIOLA system takes that fact into account, with an optimized plan for implementation projects to maximize the investment. An integrated management approach such as project portfolio management ensures that the project opportunities are effectively taken into account, that the best use is made of available resources by applying them to the highest priority opportunities, that the projects are regularly assessed, and that management actions are taken to keep them aligned with objectives.

4.4. Solution Proposal

Once IT shortfalls are identified and classified, one or more solutions are proposed from solution catalogs which provide information on various IT and/or business solutions. The source of the solution catalogs are solution and service providers. The solutions may be prefabricated ones which can be deployed with relatively minor configuration and customization. Alternatively, the solutions can be designed for the specific IT shortfall and composed accordingly by using enabling technologies. Recently, more and more solutions are composed by using Web services [23] for their cost-effectiveness. The proposed solutions, prefabricated or composed, will address the shortfalls and support the intended business transformation. For example, the client's shortfall is a gap in a business component associated with, say, marketing. The user will want a CRM (Customer Relationship Management) solution that will replace the current manual work to improve the component's performance. If a duplication shortfall is identified, the user will propose an IT consolidation solution to fix it.

VIOLA allows the user to explore the solution space to identify one or more solutions that may address one or more shortfalls of interest. The discovery of solutions for supporting components associated with a shortfall can be automatically conducted by executing a semantic query that correlates solutions and components by using their relationships to business activities. In addition, VIOLA allows the user to manually correlate them, if desired. If there is no prefabricated solution available from existing solution catalogs to support a certain hot component and/or an IT shortfall, the VIOLA system helps the user start composing a new solution, by providing a link to a solution composer tool, such as IBM's WebSphere Business Modeler, which utilizes and supports service-oriented architecture.

5. SnoBase Ontology Management System

In recent years, an increasing range of business software systems need to engage in a variety of ontology management tasks, including the creation, storage, query and integration of ontological information, and the needs turned into requirements for middleware known as ontology management systems. To make an ontology management system fit well into the current software development environment and reduce rather than increase the burden on software architects, programmers and administrators, we synthesize concepts familiar to software developers with ideas from the semantic Web and ontology communities. The SnoBase system supports the ontology needs of applications in a similar way a database management system supports the data needs of applications, by design. For programmers, SnoBase provides a Java API referred to as Java Ontology Base Connector (JOBC), which is the ontological equivalent of Java Data Base Connector (JDBC). JOBC provides a simple-to-use but powerful mechanism for application programmers to utilize ontologies without dealing with the details of ontological information. In addition, the ontology management system supports a number of query languages. At present, SnoBase supports a variant of SPARQL Query Language for RDF [20] as ontological equivalents of SQL of relational database systems.

One of challenges in the design of an industry-strength ontology management system is the versatility of Application Programming Interfaces and query languages for supporting such as diverse applications. This objective requires a careful design to simultaneously satisfy seemingly conflicting objectives such as being simple, easy-to-use for the users, and easy-to-adopt for the developers. Another challenge in ontology management is to provide ontology-enhanced industrial applications with a system that is scalable (supporting thousands of simultaneous distributed users), available (running 365x24x7), fast, and reliable. These non-functional features are essential not only for the initial development and maintenance of ontologies, but also during their deployment.

To provide a holistic management support for the entire lifecycle of ontological information, including ontology creation, storage, search, query, reuse, maintenance and integration, an ontology management system needs to address a wide range of problems; ontology models, ontology base design, query languages, programming interfaces, query processing and optimization, federation of knowledge sources, caching and indexing, transaction support, distributed system support, and security support, to name a few. While some of these areas are new challenges for ontology management systems, some are familiar and there have been active studies, particularly in relation to traditional studies on knowledge representation, or

recent studies on semantic Web standards. Our approach to the ontology management support is a pragmatic one, that is, we identify missing pieces in this picture, and engineer and synthesize them with prior work for providing a holistic management system for ontological information.

Figure 4 shows a schematic overview of the SnoBase ontology management system. Conceptually, the application programs interact with the JOBC API that provides high-level access to ontology resources and the ontology engine. The application program interacts with the JOBC API that provides an access to an implementation of the API via an ontology base driver. In this case, our driver is the SnoBase driver. In this section, we will describe the each component of the SnoBase ontology management system.

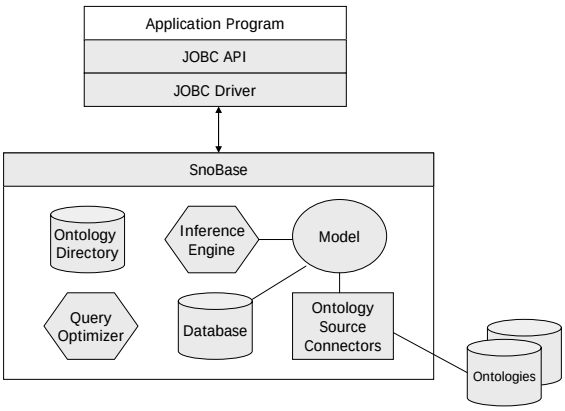

Figure 4 SnoBase ontology management system architecture

5.1. JOBC API

The SnoBase system provides a Java API referred to as Java Ontology Base Connector (JOBC), which is the ontological equivalent of the Java Data Base Connector (JDBC). The JOBC API follows the design patterns of JDBC, with several alterations. Just like JDBC, JOBC provides a connection-based interaction between applications and ontology sources. Also, JOBC provides JDBC-style, cursor-based result sets for representing query results. The similarity of JOBC to JDBC was a design decision to help application developers of SnoBase can quickly learn the programming style of JOBC from their previous experience of the popular JDBC protocol. One difference between JOBC and JDBC is that JOBC allows connections to be made without reference to a particular base ontology. Such connections provide an access to default ontologies of the top-level definitions of XML-based ontology languages such as OWL, RDF, RDF Schema and XML Schema. These definitions are required in order to process any ontological information.

5.2. SnoBase Driver

This component is an IBM driver for the JOBC interface that is equivalent to the IBM DB2 driver for JDBC. The SnoBase driver consists of Java classes that will provide an implementation of the JOBC API, and contains of a number of components: a local ontology directory, an inference engine, a working memory, a query optimizer and a set of connectors, and other infrastructure needed to support ontology management.

5.3. Ontology Directory

This component provides the meta-level information about ontologies that are available to the SnoBase driver. By default, the ontology directory contains the references to the top-level definitions of OWL, RDF, RDF Schema, XML Schema, and similar definitions for the set of XML-based ontology languages supported. In addition, the ontology directory provides metadata such as deployment information and additional sources of ontology information. For each ontology source, the directory will need to store the URI, but may additionally store information about the contents of the ontology source to aid in query optimization.

5.4. Inference Engine

This component provides a mechanism for interpreting the semantics of an ontology language, represented as a set of language specific rules. The rules are used to answer queries, when the requested fact is not immediately available, but must be inferred from available facts. For example, if the application requests the childrenOf an individual, but the working memory only contains parentOf relations, the inference engine can use the inverse property statements about childrenOf and parentOf to identify the correct response. The details of this component, different approaches to implementing this component and issues of the scalability and performance will be discussed in Section 6.

5.5. Query Language

Currently, the SnoBase system supports a variant of SPARQL Query Language for RDF as an ontological equivalent of SQL. SPARQL is a language and protocol supporting agent-to-agent query-answering dialogues using knowledge represented in RDF. It precisely specifies the semantic relations among a query, a query answer, and the ontology base(s) used to produce the answer. It also supports query-answering dialogues in which the answering agent may use automated reasoning methods to derive answers to queries. An SPARQL query contains a query pattern that is a collection of RDF sentences in which some literals and/or URIs have been replaced by variables. A query answer provides bindings of terms to some of these variables such that the conjunction of the answer sentences – produced by applying the bindings to the query pattern and considering the remaining variables in the query pattern to be existentially quantified – is entailed by a knowledge base (KB) called the answer KB. This design provides a simple but expressive query model. To make a query, a program simply describes the concept it is searching for, indicating with variables which aspects of matching concepts it is interested in receiving as part of a reply. This query model is similar to the concept of query-by-example, but with the advantage that the ontology language allows a richer method for describing the examples.

5.6. Query Optimizer

For applications that connect to large databases and/or ontologies, it will not be feasible to load the entire set of available information into working memory. Instead, the driver will query the ontology source for appropriate information as it is needed. In addition, the task of the query optimizer is to not only optimize the retrieval of information from ontology sources, but also coordinate queries that span multiple sources.

5.7. Ontology Source Connectors

These connectors provide a mechanism for reading, querying, and writing ontology information to persistent storage. The simplest connector is the file connector that is used to store information to the local file system. In addition, there will be connectors for storing ontological information in remote servers. Also, the connectors are used to implement caching of remote information to cache the definitions of the top-level ontology definitions OWL, RDF, RDF Schema, and XML Schema to allow the system to work if the W3C Web site were inaccessible.

6. Java Ontology Base Connector

As described earlier, we designed the JOBC API for SnoBase as an ontological equivalent of JDBC. The API is implemented using the abstract factory pattern [5]. An abstract factory class defines methods to create an instance of each abstract class that represents a user interface widget. Concrete factories are concrete subclasses of an abstract factory that implements its methods to create instances of concrete widget classes for the same platform. The DataManager class provides a method that is used to construct a connection, based on the URI used to initiate the connection. There is a mechanism in the DataManager that uses the database type specified in the URI to identify and load the correct driver. This driver is then used to create a connection of the appropriate type. The connection then acts as a factory to produce objects, such as statements. The objects created implement interfaces defined in the JDBC package, but have implementations that are provided by the driver that is loaded. We follow a similar design pattern in the implementation of JOBC, with several alterations, as described above. The following code sample illustrates the use of JOBC.

```
/* We connect to an ontology resource. */
Connection connection = DriverManager.getConnection();
RDFResource john = connection.createRDFResource("John");
RDFProperty isA = connection.createRDFProperty("isA");
RDFClass researcher = connection.createRDFClass(“researcher");

/* We assert a statement in inference engine: John isA researcher. */
Statement statement = connection.createStatement(John, isA, researcher);
connection.assert(statement);

/* We create a simple query. */
StatementCollection query = connection.createStatementCollection();
query.addStatement(statement);
ResultSet resultSet = connection.select(query);
```

Figure 5 JOBC Statement

In this example, the code first gets a connection. This connection is then used to create resources and a statement (john isA researcher). This statement is then asserted into the inference engine. The code then creates a simple query for the asserted fact and retrieves it from the working memory of the inference engine. More complex queries can be implemented using variables. For example, the following query requests all who are researchers.

```
/* We form a query: show me all who isA researcher. */
Variable X = connection.createVariable("?X");
Statement queryStatement = connection.createStatement(X, isA, researcher);

/* We create a simple query. */
query.addStatement(queryStatement);
resultSet = connection.select(query);
```

Figure 6 JOBC query

The results of such a query are a set of triples that include the binding(s) of the variable(s) in the query. In this case, the variable X is bound to john. Using these basic APIs, SnoBase programmers can build more complicated queries. For example, a query may contain multiple variables and multiple (query) statements. Also, note that SnoBase does not simply retrieve information previously stored for queries. Instead, by using an inference engine, it infers for answering facts that are not immediately available.

7. Model-Driven Approach to a Semantic Toolkit

Until now, we have focused on the description of the application programming interfaces, query languages, and inference engines of the SnoBase Ontology Management System. In this section, we will describe its environment for application and model development and transformation, which is equally important in the adoption of the semantic technology in the

industry. The work presented in this section is a result from collaboration between IBM T. J. Watson Research Center and IBM China Research Lab.

Participating in a number of real-world applications by using the SnoBase Ontology Management System, we have learned that it is critical to provide a comprehensive development environment including supporting tools for the application developers. A pick-and-choose approach to the best of the breed tools from different environments does not always work well for the majority of the developers and often results in a longer learning curve for the developers. A comprehensive ontology development environment often means a tight integration of various tools for application development, ontology development, model import and transformation, among others. Semantic markup languages such as W3C's RDF and OWL are based on the work in the logic and AI (Artificial Intelligence) communities, such as Description Logic and Knowledge Representation. The syntax of these languages is less intuitive to those trained for object-oriented programming and simple XML-based languages. This deficiency makes the job of subject matter experts and application developers difficult, and often affects negatively to the adoption of the semantic technology in the industry. An effective ontology application development environment should bridge this gap between the semantic markup languages and the object-oriented programmers by providing a tight and seamless integration.

Another consideration for the industry adoption of the semantic technology is the interoperability of the semantic markup languages with the well-established and widely-accepted industry standard modeling languages such as Entity-Relation (ER) modeling, XML Schema, and Unified Modeling Language (UML). The fact is that enterprises developed models in these languages for the past few decades and invested significantly to build systems around them. Despite all the advantages the semantic technology brings in, it is highly unlikely that the enterprises abandon the legacy systems and develop new systems around the semantic technology only. Rather, the users of the semantic technology in the industry would be interested in the interoperability of the modeling languages, and the reuse of the existing models and data with the semantic technology.

To address these practical requirements of the industry, we took an approach based on the Model Driven Architecture (MDA), which enables developers and users to design, build, integrate and manage applications throughout their lifecycle, while separating technology and business concerns [12]. The Object Management Group's MDA specification provides means to organize and manage enterprise architectures supported by automated tools and services for both defining the models and facilitating transformations between different model types. It also provides an open, vendor-neutral approach against the challenge of interoperability. It facilitates efficient use of models in the software development process and reuse of best practices when creating families of systems.

For implementation, we utilized the Eclipse Modeling Framework (EMF), which is IBM's open source MDA infrastructure for integration of modeling tools [4]. A model specification described in various modeling languages including XML Metadata Interchange (XMI) language, XML Schema, and annotated Java source can be imported into EMF. Then EMF produces a set of Java classes for the model, a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor. In its current implementation, EMF does not provide formal semantics definitions, inference and the related model

specifications. We are adding this capability to EMF for the comprehensive ontology application development environment and the dynamic application integration.

For adding the semantic model transformation capability to EMF, we utilized the OMG's specification of Ontology Definition Metamodel (ODM) [3], which provides metamodels of W3C's RDF and OWL in UML. By using EMF and ODM, we generated a foundational memory model, i.e., Java classes, for the constructs of RDF and OWL. This foundational memory model is referred to as EODM (Eclipse Ontology Definition Metamodel). By adding several necessary helper classes and methods to EODM, we can use it to create, edit, and navigate any models in RDF and OWL.

We also added an RDF/OWL parser to EODM, which can load RDF/OWL files into EODM and generate RDF/OWL files from EODM, i.e., serialize EODM models to standard XML RDF/OWL files. The parser utilizes an XMI adaptor which enables the transformation between the RDF/OWL models and EMF Core (Ecore) models [4]. The transformation is made possible by defining a mapping between RDF/OWL and the Ecore metamodel. The transformation opens a way to interoperability between RDF/OWL models and other EMF supported models, which currently include ones defined in XML Schema, UML and annotated Java classes. The support of other models such as Entity Relationship models in EMF will be provided in the near future. By leveraging the RDF/OWL parser and the bi-directional transformation between the RDF/OWL models and the Ecore models, ontology application developers can develop ontologies using their favorite model building tools, import them into EMF, transform their models into OWL ontologies, enrich them with semantics, leverage their inference capability, and utilize the comprehensive development facility of Eclipse and EMF.

To be more specific, the EODM Ecore model is the core model that represents ontologies in memory. It is the intermediate model for imported and transformed legacy models, as well as the generated ontology, Java code, Java editor and Java edit. The development environment allows its users to manipulate EODM Ecore models, enrich it with semantic specification, and generate Java code. A default set of mappings between metamodels of legacy models and OWL are developed in EMF. Eclipse plug-in developers can extend the mappings to handle other types of legacy models, or other elements in legacy models specifying semantics. In the generated Java code, a small foot-print inference engine is shipped with the code and can be invoked by applications. The generated Java editor and Java edit provide ready-to-use visual tools to populate or manipulated instances of OWL models. The visual tools are actually copies of the standard methods of supporting application development in EMF.

8. Concluding Remarks

An increasing range of business software utilizes ontology that externalizes knowledge for a variety of purposes in a declarative way. In this paper, we presented a semantic-based, model-driven business transformation as an emerging application domain of semantic models. While semantic technologies are actively applied to traditionally well-known areas such as information search and integration, their application to the areas of business transformation and services is still in its infancy. The presented approach is comprised of four modeling elements. First, *model-driven business transformation* provides a multi-layer model

linking business and IT semantics, and enables IT and services to accurately reflect and be driven by business value. The upper layers of model represent business semantics in the terms familiar to business executives, business managers and analysts such as key performance indicators, operational metrics, business processes, activities and governance. The lower layers of model represent IT architecture comprising a wide range of services implemented in IT infrastructure such as service-oriented architecture. Second, *component business modeling* provides a strategic-level business view of an enterprise in a dashboard, and enables business analyses based on business impacts. The CBM methodology enables a number of qualitative business analysis for identifying "hot" components and IT shortfalls that are associated with business pain points. Third, *value modeling* specifies multiple levels of key performance drivers, operational metrics and value drivers, supports various quantitative business analyses including sensitivity analyses, and enables business optimization and risk assessment. Finally, *semantic business modeling* put together business components, business activities, performance drivers and IT by capturing their relationships. It formally represents meaning of business components, metrics, and their relationships and enables automated reasoning to identify dependencies and causality relationships across business entities. By using a research prototype, we demonstrated how ontologies and semantic technologies can help infer both direct and indirect causalities of any business pain points and recommend appropriate solutions to fix business or IT shortfalls in the process of business transformation.

In the second part of this chapter, we presented an enterprise-scale ontology management system which provides functionality, scalability and performance that enterprise applications such as the proposed model-driven business transformation would demand. The primary objective of ontology management systems is to provide holistic control over management activities for ontological information by externalizing them from application programs. Ontology management systems provide ontology independence to applications in a similar way that database management systems provide data independence. One of the pragmatic challenges for ontology management system research is how to create missing component technology pieces, and to engineer them with existing results from prior research work for providing a holistic management system. We described the design and implementation of the SnoBase ontology management system, which was developed at IBM T. J. Watson Research Center. The programming interface of the SnoBase system provides a Java API, Java Ontology Base Connector, which is the ontological equivalent of JDBC. Similarly, this system supports a variant of SPARQL query language as our ontological equivalent of SQL.

References

[1] "History of Particle Physics," http://particleadventure.org/particleadventure/other/history.
[2] APQC, "Process Classification Framework," http://www.apqc.org.
[3] D. T. Chang and E. Kendall, "Metamodels for RDF Schema and OWL," http://www.sandsoft.com/edoc2004/ChangRDFS&OWLMDSW.pdf.
[4] Eclipse, "Eclipse Modeling Framework," http://www.eclipse.org/emf/.
[5] M. Grand, Patterns in Java, Volume 1, A Catalog of Reusable Design Patterns Illustrated with UML, John Wiley & Sons, 1998.
[6] IBM, "Architecture of Business," IBM Global Technology Outlook, http://www.ibm.com, 2004.

[7] IBM, "Component Business Modeling," http://www-1.ibm.com/services/us/bcs/html/bcs_componentmodeling.html.
[8] J. Lee, "Model-Driven Business Transformation and Semantic Web," Communications of ACM, December 2005.
[9] J. Lee, and A. Ivan, "Value-Centric, Model-Driven Business Transformation," IEEE Joint Conference on E-Commerce Technology (CEC '06) and Enterprise Computing, E-Commerce and E-Services (EEE '06), San Francisco, California, June 26-29, 2006.
[10] J. Lee, and R. Goodwin, "Ontology Management for Large-Scale Enterprise Systems," Journal of Electronic Commerce Research and Applications, Vol. 5, No. 3, 2006.
[11] D. McDavid, "The Business-IT Gap: A Key Challenge," IBM Research Memo, http://www.almaden.ibm.com/coevolution/pdf/mcdavid.pdf.
[12] OMG, "Model-Driven Architecture (MDA)," http://www.omg.org/mda/.
[13] OMG, "Universal Modeling Language (UML)," http://www.omg.org/uml/.
[14] Y. Pan, G. Xie, L. Ma, Y. Yang, Z. Qiu, and J. Lee, "A Model-Driven System for Ontology Engineering," Journal of Data Semantics VII, 2006.
[15] Topic Maps, http://www.topicmaps.org/.
[16] W3C, "Extensible Markup Language (XML)," http://w3c.org/XML/.
[17] W3C, "RDF Vocabulary Description Language 1.0: RDF Schema," http://www.w3.org/TR/rdf-schema/.
[18] W3C, "Resource Description Framework (RDF)," http://www.w3.org/RDF/.
[19] W3C, "Semantic Web," http://www.w3.org/2001/sw/.
[20] W3C, "SPARQL Query Language for RDF," http://www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/.
[21] W3C, "Web Ontology Language (OWL)," http://www.w3.org/2004/OWL/.
[22] W3C, "Web Services Architecture," http://www.w3.org/TR/ws-arch/.
[23] W3C, "Web Services," http://w3c.org/2002/ws/.
[24] W3C, "XML Schema," http://w3c.org/XML/Schema.