A Bottom-Up Strategy for Enterprise Ontology Implementation

Sang-goo Lee[1], Taehee Lee[2], Dongkyu Kim[2], Jonghoon Chun[3]

[1] Center for E-Business Technology, Seoul National University, Seoul, Korea

[2] Prompt, Inc., Seoul, Korea

[3] Myongji University, Kyungkido, Korea

Abstract The benefits of semantics for intelligent and interoperable services have been widely accepted in the computing community. Semantics can also improve the quality of software systems that deal with the every day operations of an enterprise. However, building and utilizing an ontology in the enterprise environment is very difficult and risky since ontology inference is complex, slow, and often unpredictable. Also, ontology management is not scalable and building an ontology is a complicated process that takes a huge amount of resources. Presented is a bottom-up strategy for large scale ontology implementation in commercial settings. Important components of the strategy include harvesting instance level ontology, creating success cases early, and ensuring scalability.

Keywords ontology, semantic technology, enterprise systems,

1. Introduction

One of the main challenges in building enterprise applications has been to balance between general functionality and domain/scenario-specific customization. The lack of formal ways to extract, distill, and standardize the embedded domain knowledge has been a barrier to minimizing the cost of customization.

In [14], *information finding* and *information integration* are presented as the two main challenges for the field of information technology. Although statistics based keyword searches are getting better, the computer system still does not *understand* the information it is finding for us. For example, how do we tell the computer to find me a dentist who is available next Tuesday afternoon in the southern area of Seoul? As for information integration, can a computer system automatically recognize that the 'balance' attribute in one bank's database is the equivalent of the 'amount' field in the database of another bank? These are still very hard problems and the missing piece in this puzzle is *semantics* [14]. If the computer system is able to comprehend the semantics behind the data and entities it processes each day, it would be able to provide us with a set of services that are far more intelligent.

Semantic technology refers to the area of information technology that deals with the creation and management of semantics separately from data, content, and program code. It provides means to discover, manage, reason with, and utilize meanings. Separately managing the semantics of an application domain provides an opportunity to analyze domain knowledge, make domain assumptions explicit, separate domain knowledge from operational knowledge, provide common understanding of the information structure, and enable reuse of domain knowledge [12]. Applications of semantic technology include the Semantic Web [2], context-awareness, intelligent search and match, and information integration.

At the core of semantic technology lies

ontology. Ontology is a formal specification of concepts of the domain of interest [4]. Ontology provides a reference domain model that both human and software can refer to for various purposes such as search, browsing, interoperability, integration, and configuration [11]. Computer programs may utilize the ontology to infer semantic implications of a keyword or a query result.

Research has been very active in this area in recent years, especially in the context of W3C's Semantic Web [2] including the efforts for building a foundation for ontology (e.g., standard semantic markup languages such as RDF [9] and OWL [3]). Even with the rich amount of research in ontology, there are still gaps to be filled in actual deployment of the technology/concept in a real life commercial environment. The problems are hard especially in those applications that require well-defined semantics in mission critical operations.

In this paper, we present some of the challenges in building and utilizing an ontology for commercial use and present a bottom-up strategy that addresses some of these challenges.

2. Ontology Implementation

Ontology implementation is a very difficult task. The myriad of technical standards and specifications only address the formats of ontology. For example, RDF and OWL specify the syntax for how certain concepts and relationships should be represented but do not tell us whether 'rock & roll' and 'music' are related through a relationship called 'genre'. In other words, the technical standards provide us with the tools (pen and paper) to write a book, while the formidable task of actually writing the book, i.e., filling the blank papers with contents, is left to the ontology implementer. Too many ontology projects have failed to live beyond proof-of-concept demonstrations because a pragmatic system would require a huge amount of investment in building the contents of the ontology. For example, with over 21 years, 750 person-years, and 75 million dollars, Cyc [10] hosts some 3,000,000 hand-entered facts and rules. The knowledge base has provided a number of compelling demos and presentations but the jury is still out on whether an enterprise level application can be built on it.

Inference is an essential aspect of ontology which refers to the ability to extract new information about a concept by searching and following the relationships in the ontology. For example, given a class *A* and ontology *O*, the task to find the lowest class *B* in *O* that subsumes *A* is a basic inference (called classification in description logics [1]). Reasoning in the level of concepts and classes is called T-box reasoning and reasoning that involves instances of classes is called A-box reasoning. Inference is inherently NP-complete or harder, and most of the inference-oriented ontology systems deal with T-box reasoning and do not scale well to A-box reasoning as the number of instances grows.

Other issues that must be addressed in ontology implementation are listed below [5, 8].

- **Modeling**: Level of abstraction problem haunts all aspects of ontology design. Multiple views and taxonomies, often with conflicting semantics, present another challenge for the field engineer.
- **Ontology – DB Integration**: The ontology can be modeled as meta data for the database, where the database alone represents the information content of the system and the ontology is a secondary facility. On the other hand, the ontology can be modeled as an integral part of the database, in which case, ontology must be part of all queries and operations. Trade-off includes implementation complexity, semantic richness, and efficiency.
- **Ontology Lifecycle Management**: Populating the ontology is a daunting task which can make or break the project. The job is complicated by multiple formats, semantic mismatches, errors or dirty data in pre-existing information sources. Change

management (versions, mergers, decompositions, etc.) is another complicated issue.

- **Accountability and Control**: One of the biggest concerns inhibiting ontology adoption in enterprise applications is its lack of control. When is an ontology complete, in the sense that it holds sufficient content to support all mission critical operations? Is the behavior/performance predictable?
- **Human Factors**: Building and maintaining the ontology requires much more than software engineers. Domain experts must define the concepts and relationships of the domain model. Ontological information model is not a concept easily understood by non-computer/ontology experts. A set of intuitive guidelines must be provided. Easy-to-use tools are also essential.

3. Issues in Business Implementation of Ontology

We first list some of the business applications that can potentially benefit from the use of ontology.

- **Mobile Search**: With the limited bandwidth and input/output features, even a marginal level of intelligence can enhance the user's search experience in a mobile environment. Simple synonym extensions and location-time based context filters can be used.
- **Personalized Contents Offerings**: Collaborative filtering and other personalization and recommendation algorithms [13] can be enhanced by adding semantics to data pertaining to items (products) and customers.
- **Location Based Service**: One of the key applications of context-aware systems is location based service where the system provides contents and services according to the context of the user location. Context is a highly semantic feature.
- **Information Service**: Intelligent information services need to be personalized and context-sensitive. The presentation of information can be enriched by supporting multiple taxonomies for browsing.

We have observed that most practical enterprise applications including the set of services listed above would need only a relatively simple concept hierarchy and do not need the full scale inference capabilities of the ontology languages. For example, an ontology for location based service would not need thousands of classes and relationships but requires only a limited degree of inference to navigate through the relationship links.

Several critical factors contribute to implementing a service that is intelligent and also makes practical business sense. First, a deep and accurate understanding of the transactional data is essential. The most effective class of recommendation algorithms in personalization services is collaborative filtering algorithms, and these require careful analysis of the transactional logs. The co-occurrence of certain pairs of values in transactions implies the existence of an ontological relationship between the values. Using simple data mining techniques, we can extract a considerable amount of relationships from the existing database.

Second, a standardized and clean database is a necessary component for any type of quality service. The simplest form of ontology is a glossary [12]. As simple as it is, a standard glossary can be very effective in enhancing the quality of an information system. The column names of a database as well as the values in the records are all targets of control. Clean and standardized data is as important as a complex inference engine for intelligent services.

Third, a simple but effective rule processing engine is required for executing the intelligent service. An expert rule stating "high income professionals are likely to be interested in stock related contents" is not an ontological rule that can be represented in OWL. However, such rules are key ingredients of a personalized service and have been used in CRM (customer relationship

management) systems. Either the ontology (including its inference) model must be extended to accommodate such rules or seamless integration between the ontology system and the rule processing system must be provided. Also, the complexity of the inference must be tractable and controllable.

Forth, an enterprise ontology system must promise adequate performance. The problem with most of the current ontology query processing systems is that they are in-memory based, i.e., rules and facts must all fit in main memory. For a moderately large set of ontology, these systems either crash or take too much time to be usable. We need a different approach for performance and scalability.

4. The Bottom-Up Strategy

The current research efforts in ontology focus more on theoretic aspects such as technical specifications of OWL and the inference mechanism. Although these are foundational works that cannot be done without, we need to quickly build success stories in actual deployments. We propose in this section a bottom-up strategy that allows for fast deployment of ontology based services. The strategy is a result of our experience in building an operational product ontology system for a government procurement service [7, 8]. The system is designed to serve as a product ontology knowledge base; not only for the design and construction of product databases but also for search and discovery of products. Especially, the keyword-based searching over product ontology database demands different techniques from those over conventional document databases or relational databases, and was designed to reflect particular characteristics of an ontology.

4.1. Building the Ontology

If we classify an enterprise ontology into *instance level ontology* and *concept level ontology* (roughly corresponding to the T-box and A-box in DL), the instance level volume would make up for more than 90% of the whole ontology (see figure 1). We propose to build the ontology bottom-up from the instance level. The justification is that it makes sense to utilize the vast amount of information residing in the enterprise database. Furthermore, this approach generates results instantly. In addition to the extracted ontological features, the process provides detailed analysis of the current database state, which is an invaluable piece of information for database administrators.

The instance level ontology is built in a batch fashion by bulk-loading and transforming the data from the existing records in the database. Since entities that have references to other entities should be built later than the referenced entities, transformations should be performed in a specific order to preserve the reference dependency of each entity. Since the automatically constructed ontology may not show the quality that the domain experts expect, methods for improving the quality are required.

If a pattern occurs consistently in the instance level, it is a candidate for generalization into a concept level entry. For example, if all 46” TV sets have feature ‘widescreen’ then it can be conjectured that 46” is the size available only for widescreen TVs. This may be true or may only be a coincidence so domain experts should be involved in the generalization process. Such generalized piece of knowledge makes up the set of *publishable domain ontology* in figure 1. It is the class of ontology at the concept level that can be inferred from the instance level.

Thus, the instance level ontology and the publishable domain ontology which is part of the concept level ontology are built bottom-up from the legacy databases. Then on top of these, we can always utilize general concept level ontology that can be imported from other sources such as Cyc.

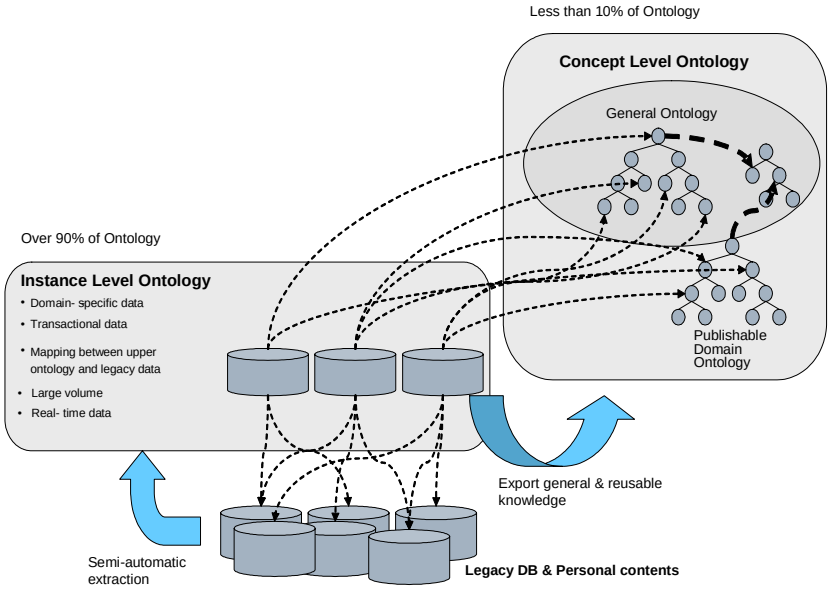

Figure 1. Bottom-up construction of an enterprise ontology

4.2 Inference

As briefly mentioned previously, there are various types of rules that are used in an enterprise computing environment and not all of them can be expressed in ontological languages. Consider the case of a CRM system for a consumer electronics retail store. The ontological knowledge that 46" TV sets are always widescreen TVs can be helpful in correctly identifying the customers who purchased widescreen TV sets (assuming there are 46" TV sets that are not annotated as widescreen). So, it makes sense to somehow integrate (or at least consolidate) the CRM knowledge base with the ontology. Since both knowledge bases have rules, we need to carefully identify the roles and characteristics of the various types of rules for consolidation.

First, there are rules that are part of the ontology. These are the rules that constrain the relationships or cardinality of ontological entities. Inference, such as subsumption and instance checking in description logics, must be supported to utilize these rules.

The second type of rules is the if-then-else pattern rules used in conventional rule-based system such as CRM. An example would be "if person *x* is a female and has a professional job then there is a 30% chance she enjoys concerts." Such rules are often extracted as a result of data mining exercises. One might argue that it is possible to represent this piece of information in OWL. While it is true, we believe doing so serves no practical value since the CRM system is tuned to process and utilize this type of rules much more efficiently than a general ontology inference engine.

The third type of rules is action rules such as triggers and marketing operation directives. An example is "if person *x* purchases product *y*, then push product *z* on her screen." Again this type of rules can be found in CRM systems and marketing campaign management tools.

We propose to separate the rules according to their use and have different systems manage each class of rules. During the ontology construction phase, it would be worthwhile to re-examine the rules in other systems and regroup the rules according to their roles and features. Each group of rules can then be stored and managed by the respective subsystem that is designed and tuned for the specific purpose. An example of rule consolidation and corresponding subsystems is shown in figure 2.

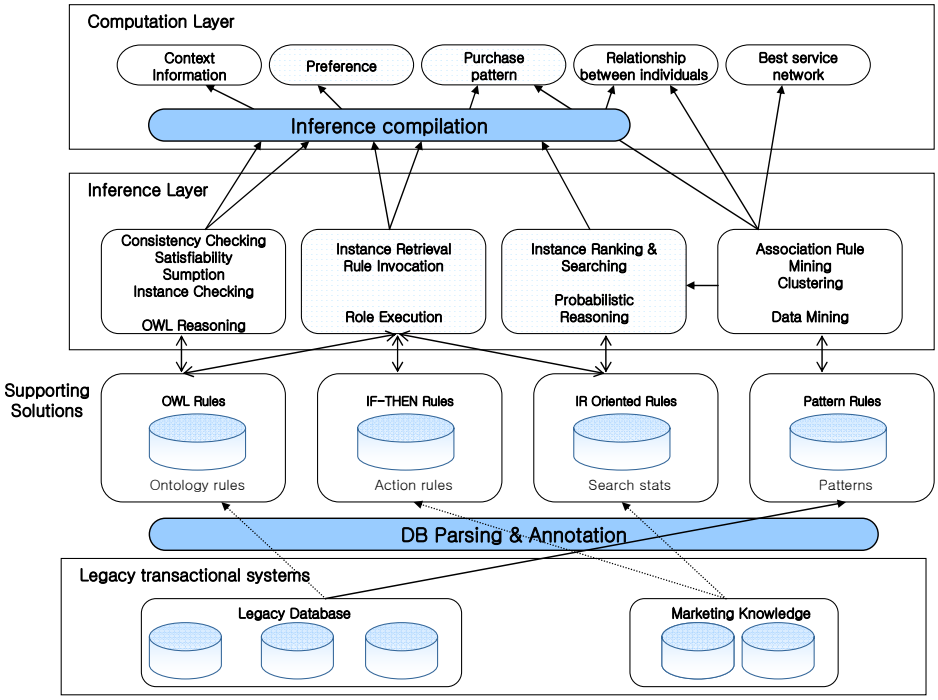

Figure 2. Consolidation of rules and subsystems

4.3. Scalability

Our goal is not only to design a 'conceptual' ontology model but also to implement it as an operational ontology database. One way to achieve this goal may be through using an ontology language such as OWL and building an OWL knowledge base that represents the intentional and extensional concepts and relationships for the ontology. This approach, mainly favored by the research community, may be beneficial for integrating the domain ontology model with an inference engine for the language.

However, it is technically too complicated to represent and comprehend the domain for a domain expert who has little knowledge in the formal language. More importantly, from a practical point of view, there is no publicly known robust engine to manage a large knowledge base with practical performance.

For example, our product ontology database contains over 700,000 item level products and more than 900,000 concepts including product classes, attributes and unit of measures (UOM). Concepts are linked by semantic relationships and there are more than 21 million semantic links representing these relationships. Needless to say, the size of the database keeps growing every day. As for reasoning, we only needed a rather limited-set of reasoning capabilities such as transitivity and inverse relationships. Naturally, the general purpose reasoning capability of an OWL engine was considered an over-kill.

An alternative way is to build an ontology database on a commercially operational database system such as a relational or object-relational DBMS. This way we can take advantage of existing standards for data management and the DBMS features that have been optimized over the years in terms of robustness, scalability, and performance. Since a DBMS by itself does not support reasoning functions, the set of reasoning capabilities must be implemented within the ontology applications. Even so, we believe that this approach is a sure way to make an ontology

database operational.

Table 1, borrowed from [6], shows the key differences between these two comparative methodologies. In order to balance between these two extreme goals and characteristics, we need an adaptive approach which assures scalability and extensibility.

Table 1. Model implementation methodology

	OWL-based knowledgebase approach	**Relational database approach**
Theoretical background	OWL, DAML+OIL, Topic Maps, Description Logics, FOL, …	EER, Table, SQL, Relational Algebra & Calculus, …
Pragmatism	Theoretical	Operational and practical
Ontology representation	Rich for various semantic constraints	Rather limited for complicated semantic constraints
Ontological Reasoning or inference	Supported by OWL reasoning engine. Reasoning complexity may be high.	Limited. Reasoning capabilities coded within applications.
Commercial level	Publicly no engine is available to support a large knowledge base.	Many DBMSs are commercially available.

We propose to build an ontology database on top of an operational database system and, at the same time, provide an exporting mechanism from the database to an OWL knowledge base. In other words, each modeling construct in an object-relational database can be translated into the corresponding OWL representation. Then a set of translated representation may form an OWL knowledge base, and the ontological query and reasoning capabilities of an OWL engine could be exploited. In fact, within our project, we have developed a module that translates relational tables representing concepts such as products, classification schemes, attributes and UOMs (unit of measure), and their relationships into OWL representations.

4.4 The Ontology Manager

The ontology manager is the software system responsible for all aspects of the ontology; from construction of the ontology and interoperation with other systems within the enterprise to evolution and change management. The ontology manager should support the bottom-up approach described in this section. A simple sketch of its architecture is shown in figure 3.

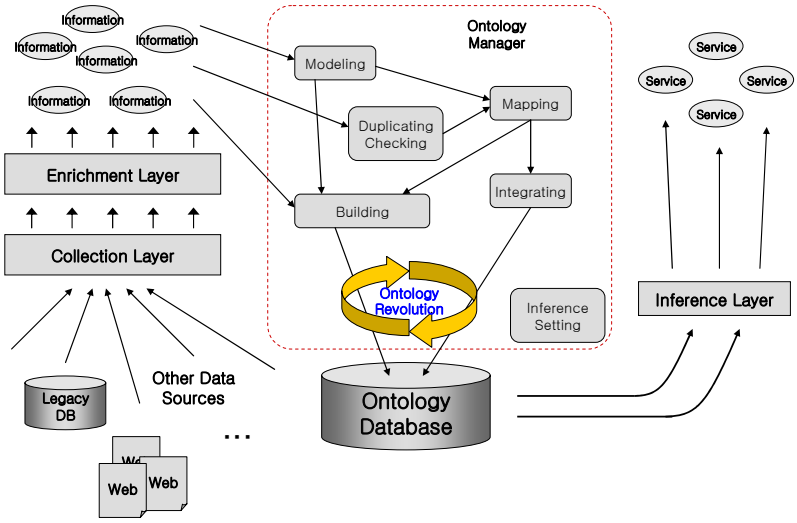

Figure 3. The ontology manager

5. Conclusion

Semantics is essential in providing intelligent context-aware services. Ontology is the core component of providing semantics to an information system. Building an ontology that is robust and efficient enough to support commercial operations is a formidable challenge.

We have presented a bottom-up strategy for this challenge. It is based on our experience on building a large scale instance level ontology that is currently being used in a public setting. This is part of an ongoing effort, i.e., details of the strategy is constantly refined at this moment. What we have presented here represent the philosophy that underlies our approach.

We are currently applying our methods to a digital content service environment where the usage logs and tags and blogs are our target for instance analysis.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider (eds.), *The Description Logic Handbook*, Cambridge University Press, 2002.
2. T. Berners-Lee, J. Hendler, O. Lassila, The Semantic Web, *Scientific American*, May 2001.
3. M. Dean, G. Schreiber (Eds.), *OWL Web Ontology Language Reference*, W3C Recommendation, 2004.
4. T. R. Gruber, A Translation Approach to Portable Ontologies, *Knowledge Acquisition*, 5(2):199-220, 1993.
5. D. Kim, Y. Chang, J. Lee, S.-g. Lee, Ontological Approaches to Enterprise Applications, *23rd International Conference on Conceptual Modeling (ER2004)*, 2004, 11 / *Lecture Notes in Computer Science* (Vol. 3288, pp. 838-840).
6. I. Lee, S. Lee, T. Lee, S. Lee, D. Kim, J. Chun, H. Lee, J. Shim, Practical Issues for Building a Product Ontology System, *International Workshop on Data Engineering Issues in E-Commerce (DEEC2005)*, IEEE Society, 2005.
7. T. Lee, S. Lee, I. Lee, S.-g. Lee, D. Kim, J. Chun, H. Lee, J. Shim, Building an Operational Product Ontology System, Electronic *Commerce Research and Application (ECRA)*, Vol 5. No 1 (Jan 2006).
8. T. Lee, J. Shim, H. Lee, S.-g. Lee, A Pragmatic Approach to Model and Exploit the Semantics of Product Information, *Journal on Data Semantics VII*, pp. 242-266. 2006.
9. F. Manola, E. Miller (eds.), *Resource Description Framework (RDF) Primer*, W3C Recommendation, 2004.
10. C. Matuszek, M. Witbrock, R.C. Kahlert, J. Cabral, D. Schneider, P. Shah, D. Lenat, Searching for Common Sense: Populating Cyc from the Web, *20th National Conference on Artificial Intelligence*, July 2005.
11. D. L. McGuinness, Ontologies Come of Age.: *The Semantic Web: Why, What, and How* (D. Fensel, et al., eds.), MIT Press, 2001.
12. N. F. Noy, D. McGuinness, Ontology Development 101: A Guide to creating your first Ontology, *Stanford KSL Technical Report*, KSL-01-05, 2000.
13. B. Sarwar, et al, Item-based Collaborative Filtering Recommendation Algorithms, *ACM WWW10*, 2001.
14. H. Stuckenschmidt, F. van Harmelen, Information Sharing on the Semantic Web, Springer, 2005.