Implementing the Semantic Business Development Control Center

Dieter E. Jenz,
President, Jenz & Partner GmbH

Needed: The Way Out of the “Software Babel”

Today, the so-called “software babel” is lamented by business and IT managers all over the world. The software solutions portfolio has grown tremendously over the years, which in turn requires enormous effort to make applications “talk” with each other. While plumbing at the technical level is supported by a plethora of tools, interaction at the semantic level is much more of an issue, and a prevailing one.

Yet another problem plagues business and IT managers alike: domain experts and IT experts still do not understand each other well. Project failures due to talking in different languages, resulting in bad requirements, seem to be the rule rather than the exception.

Highly praised techniques have not really eliminated the “semantic gap.” For example, entity relationship modeling and object modeling techniques are based on a very limited vocabulary and grammar. As has been seen often, after a review meeting with stakeholders, IT people were not sure if the audience understood what they presented. Viewed from the other side, stakeholders were not sure if the IT people got it right, after all.

Graphical modeling techniques, as good as they are, share a common shortcoming: The modeler can only represent a small subset of the knowledge acquired in discussions with domain experts. What cannot be expressed with the limited vocabulary and grammar remains in the heads of the people. Attaching free-text notes is a great help, since they are not machine-interpretable.

Commercial off-the-shelf software usually comes with a repository, which is based on a proprietary meta-model. The same holds for purpose-built software, developed in-house or by a contractor. Each meta-model item is given a specific meaning. For example, the meta-model designer associates “customer” with a meaning. But what is a “customer”? It is not uncommon for a larger organization to find out that “customer” has different meanings in application systems. In other terms: “customer” means different things.

The semantic gap not only exists between business and IT people, it also exists among application systems, hence the “software babel.” And it exists among software development tools. Modelers and developers are forced to maintain definitions in multiple tool repositories. Keeping these definitions in sync is a real pain and costs a lot of money.

The “software babel” has created tool-specific “information islands,” which also result in “information bottlenecks,” meaning that knowledge is not available on a broad scale. Specific tool expertise is required to create, maintain, and interpret models. Although business and IT managers are determined to avoid dependency on relatively small groups of experts, tool vendors were obviously successful with their “information island” strategy.

A strategic, overarching approach is required, which helps overcome the “software babel” and settle on some kind of “world language.” Unimpeded information sharing with rich semantics at all levels without loss of information is a major prerequisite for shortening time-to-market of processes and applications, thus strengthening an organization’s ability to keep abreast of the competition. The Semantic Business Development Control Center, suggested in this paper, represents the implementation of a strategy that consequently takes advantage of “smart data,” eliminates “information islands,” and avoids information bottlenecks.
**Bridging the Semantic Gap**

The semantic gap has been widely discussed during recent years, but solutions have not really arrived on the desks of domain experts or IT experts. The failure of “artificial intelligence” more than a decade ago had a catastrophic effect in that it put the “academic exercise” tag on potential solutions for a couple of years.

There is demand for a single “world language,” which helps eliminate the semantic gap, and which can be understood and spoken by domain experts and IT experts alike. While “knowledge languages,” such as KIF (Knowledge Interchange Format), have long been restrained to the academic domain, a newer language seems to have the potential to become ubiquitous in the long run: The Web Ontology Language (OWL).

OWL allows people to formalize what they mean by various terms and how these terms are related with each other. In general, nouns are represented as classes, quite similar to object modeling. As such, an ontology is an information model. Yet, there is more to OWL: It also allows for the instantiation of the information model, meaning that class instances (individuals) can be created and manipulated.

**Current Initiatives**

The World Wide Web Consortium (W3C) drives forward the development of the Semantic Web, which is a collection of well-defined specifications. OWL is part of the Semantic Web architecture and is layered on XML and RDF (Resource Description Framework).

The Object Management Group (OMG) is well known for UML, the Unified Modeling Language, which has gained broad support in the object-oriented domain. Today, UML support is considered mandatory for an object-oriented CASE tool. Starting in the mid 1990s, the OMG has also begun to define other standards, such as the Meta-Object Facility (MOF) and XMI (XML Metadata Interchange). MOF, UML, and XMI are all part of OMG’s four-layer metadata architecture.

The meta-metamodel, represented by MOF, is at the top of the metadata architecture. It defines an abstract language and framework for specifying, constructing, and managing technology neutral metamodels. It forms the foundation for the definition of any modeling language, such as UML, or even MOF itself. In addition, MOF defines a framework for implementing repositories that hold metadata (e.g., models) described by metamodels. The MOF is generic enough to provide for the definition of an ontology-modeling language in MOF.

A Request For Proposal (RFP), issued by the OMG, calls for the definition of an Ontology Definition Metamodel (ODM). Ideally, the ODM is defined as a MOF-compliant language, so that ontologies can be stored in MOF-based repositories as well as shared and interchanged through XMI.

To make use of the graphical modeling capabilities of UML, an ODM should have a corresponding UML Profile to support reuse of UML notation for ontology definition. The UML Ontology Profile allows for...
graphical ontology editing. Hence, UML tools can be reused for ontology modeling, enabling users to use graphical UML syntax.

Both UML models and ODM models are serialized in XMI format; thus the bidirectional transformation between them can be performed using XSL Transformation (XSLT). As a consequence, ontologies can be shared by means of XMI, too.

Considering ongoing efforts in the OMG and W3C camps, the road will eventually lead to convergence. The OMG has already come a long way since the beginnings of the UML version 1 specification, which suffered from poor semantics. Today, clearly, the OMG is set to increase the semantics of object-oriented systems.

Although there are still a couple of rough edges, there is enough reason to suggest that financial investments in either technology will not prove lost in the long term. While XMI-based ontology transformation into a UML model has already been demonstrated, it will also be possible to transform a UML model into an ontology.

**The Cornerstone: An OWL-Based Repository**

The ODM specification is still under development, which means that there is still no standards-backed way of making use of MOF-based repositories. However, an OWL-based repository is a viable alternative, which also meets the need for protection of financial investments.

As a declarative language, OWL provides all the language constructs necessary to define an information model, which can be implemented in a repository. A repository can store definitions (e.g. class definitions) and instance data. Hence, it is not unjustified to draw an analogy to a relational database.

Although a relational database and an ontology are different things, and it seems like comparing apples to oranges, it often helps to relate a very well known technology to a largely unknown technology in order to better identify where a newer technology is particularly useful.

<table>
<thead>
<tr>
<th></th>
<th>Relational Database</th>
<th>Ontology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic concept</td>
<td>Two-dimensional tables, related with each other through primary/foreign key fields</td>
<td>Taxonomy (Vocabulary and hierarchical structure) + relationships</td>
</tr>
<tr>
<td>Inheritance</td>
<td>Not an original relational concept – supported through object-relational extensions</td>
<td>Supported (single and multiple inheritance)</td>
</tr>
<tr>
<td>Relationships</td>
<td>Relational tables are linked with each other through primary and foreign key fields (attributes)</td>
<td>Classes are linked with each other through object properties (attributes)</td>
</tr>
<tr>
<td>Nesting</td>
<td>Databases cannot be nested</td>
<td>Ontologies can be nested (an ontology may import one or more ontologies)</td>
</tr>
<tr>
<td>Data definition and manipulation language</td>
<td>Structured Query Language (SQL)</td>
<td>Web Ontology Language (OWL)</td>
</tr>
<tr>
<td>Query language</td>
<td>SQL</td>
<td>No standardized Query Language, various options, e.g., RDQL (RDF Query Language), similar to SQL</td>
</tr>
<tr>
<td>Triggers</td>
<td>A data manipulation operation may fire a trigger, which causes a “procedure” to execute</td>
<td>No, OWL is purely declarative</td>
</tr>
</tbody>
</table>
Reasoning, inferencing | Not supported | Supported through OWL
---|---|---
Separation of schema and user data | Data dictionary (system catalog) and user data are separate | Definitions (classes, properties, etc.) and instances (individuals) may or may not exist in the same file
Access granularity | Single rows, multiple rows (dependent on SQL statement) | Information model items (classes, properties, etc.), and individuals (instances)

Let us think for a moment of an ontology as a database. An ontology contains definitions (class definitions, property definitions, etc.) and may also contain data (class instances, individuals). The term “knowledge base” is generally used as a synonym to denote an instantiated ontology (structure and data). Ontology definitions loosely resemble the database schema of a relational database, while class instances loosely relate to rows in a relational database.

OWL allows us to define concepts and relationships among them in a semantically rich fashion, which makes it possible to do automated reasoning. As a simple example, we could state that drivers are persons that drive cars. If we also state that all drivers must be adult persons, it can be inferred that all drivers are adult.

Although OWL is a rich “data definition language,” an ontology will not replace existing relational databases anytime soon. That is for a simple reason: an ontology loads into memory as a whole, including all imported (included) ontologies. One might think of a puzzle, which is complete when all the pieces are in place.

An OWL file usually represents a single ontology, comprising definitions (classes, properties, restrictions) and, potentially, individuals (instances). However, there is no strict rule. It is also possible to split definitions and individuals and put them in separate files. This helps with separation of concerns. In case of separate files, the “data ontology” (which contains instances) imports the “definition ontology,” effectively preventing users from changing ontology definitions (an imported ontology is protected against modifications).

It is also possible to store ontologies in a relational database, rather than in ordinary files. At present, not all ontology editor tools support such functionality. However, enabling multiple users to work with the same ontology in parallel and make modifications, more or less mandates the use of a database as ontology container.

Current ontology editors will evolve into fully-fledged ontology management systems over time. The evolution of relational database management systems may be witnessed once again.

**The Move Towards Ontology Management Systems**

In the early days, relational database management systems were based on a single-process architecture, meaning that database access libraries were linked to the user application. Later on, the two-process architecture was introduced, resulting in a separation of user application and actual database access. The user application (the client) communicates with the database management system (the server) through a defined API and a protocol. The two-process architecture still forms the foundation of today’s database management systems.

In addition to associative access capabilities, which SQL provides, an “ontology management system” would require strong navigational access capabilities, as seen in today's object-oriented database management systems (OODBMS).

From a technological viewpoint, there is nothing that would rule out building an “ontology management system.” The definition and manipulation language exists in the form of OWL, which is now a W3C Recommendation. There are still a couple of candidates for a query language, RDQL (Resource Definition Query Language) being probably the most popular language at this time.
The fact that an ontology is loaded into memory when initially opened does not preclude multi-user access. The “ontology management system” is responsible for setting, holding, and releasing locks, so that a user cannot interfere with the work of others and possibly cause corruption. At the technical level, repositories may be physically dispersed over multiple locations. Federated repositories appear as a logically coherent entity, meaning that users perceive the repository as a single entity. Although the technology is not yet there, development is under way and results are expected in late 2004/early 2005.

However, repository federation will only be on the feature request list of some large enterprises, while the majority of organizations will be happy with a single-location repository.

The Semantic Repository Backbone

While ontologies cannot be viewed as a replacement for enterprise databases within the next few years, it is indeed possible to make powerful use of ontologies in software development and business process management at large. An ontology can serve as a semantic repository backbone in that it delivers a semantically rich, robust, and consistent information model, as well as having the capability to store actual data. As such, an ontology implements both a registry and a repository.

An organization is totally free to define its information model so that it meets its specific requirements. In contrast, for example, a database management system comes with a pre-defined information model and a data dictionary (system catalog), which cannot and should not be extended.

In addition, the semantic repository backbone can incorporate multiple ontologies, which may represent different domains of responsibility. For example, a generic business ontology may form the lowest layer. One or more industry-specific ontologies may represent the second layer, followed by an organization-specific ontology as the highest layer. These layers represent business knowledge and maintaining these ontologies would be the responsibility of business experts and business analysts. Each ontology may have a different owner.

The business view is complemented by an IT view, represented by an IT ontology. The IT ontology comprises business objects, which are representations of business entities. While business experts are used to describing real-world concepts in the form of business entities, software engineers have a different worldview. They use object-oriented analysis and design methods to create object-oriented models, which represent some “system-oriented” view in that object models are optimized for system performance.

The semantic repository backbone brings business and IT together. Relationships between business and IT concepts can be established, resulting in a mapping. A business entity (business view) may be

1 See ebXML Registry/Repository specification and the freebXML Registry Open Source Project (http://ebxmlrr.sourceforge.net). The freebXML Registry/Repository is a possible ontology container, since freebXML will provide OWL support.
represented by multiple business objects (IT view), and vice versa.

Lessons Learned

Repositories are all but a new idea. Early implementations date back into the 1970s. Vendors advocated a centralized repository model, which, however, did not gain acceptance on a large scale. The failure may be attributed to a number of reasons, such as complexity, performance issues, and immature standards. Despite these issues, the need for an integrated repository has never gone away.

Over the years, the decentralized repository model became more popular. Repositories are now packaged in development tools, database management software, and so on. However, this comes at a price: Definitions may be in conflict with each other from an overall cross-repository perspective. In addition, data interchange between repositories is not as simple as it seems in the first instance. In order to increase functionality and improve performance, many software vendors have implemented proprietary extensions.

XMI (XML Metadata Interchange), an XML-based data interchange format, specified by the Object Management Group, has been regarded as a once-and-for-all solution by many. However, as can be seen often, the issues are in the “fine print.” Not all tool vendors support XMI, and, if support is provided, it may not be for the latest XMI version. As a consequence, even the use of XMI may cause headaches and still needs to live up to its promise. Learning from experience, decentralized non-integrated repositories are not a silver-bullet solution either.

Shifting the focus away from the centralized vs. decentralized discussion to the repository language level, it becomes obvious that the road will lead to convergence of hitherto disparate technologies. MOF is used to represent UML models in a machine-processable way, while XMI works as a serialization layer for MOF. RDF (Resource Description Framework), which is the language layer underneath OWL, is similar to MOF in that it provides a generic object model. RDF acts as an XML encoding that supports a data model with node and relationship types.

Both MOF and RDF give a generic way of representing objects and relationships between them. It is indeed possible to layer UML on top of RDF. However, since no such venture is known of at this time, this is more an untapped potential. OWL, in contrast, already sits on top of RDF.

To sum up, implementing the semantic repository backbone means opting for all of the following:

- federated repositories;
- a coherent information model with rich semantics;
- a powerful repository language;
- a powerful serialization layer for data interchange.
Implementing the Semantic Business Development Control Center

With the semantic repository backbone as the foundation, the next step can be envisioned: the semantic business development control center.

Shrinking business cycles result in the demand to shorten time-to-market of IT solutions. Considerable progress has been made in the area of software development, resulting in much increased developer productivity. However, it still takes too much time and resources to define and transform business requirements into models from which software can be generated.

There is a need for a semantic business development control center, which provides the environment necessary to establish an end-to-end software process. It rests on integrated ontologies which form the semantic repository backbone, hence providing a "one world view." It brings together business experts and IT experts and bridges the semantic gap between business and IT. Business experts and IT experts express knowledge in the same language.

The semantic business development control center incorporates various tools and functions, such as an ontology editor, tools that generate artifacts from ontologies, functionality for graphical visualization of ontology content, and statistics functions. It forms the central source of information for business experts and IT experts alike, and promotes shared understanding, knowledge acquisition, and organization.

As of today, the semantic business development control center is largely a vision. The various pieces will fall into place over the next months and years as tools become available and mature. However, one of its major components, the semantic repository backbone, can be made a reality today and used effectively to create business value.

Taking Controlled Steps

By taking advantage of the capability to define the information model in accordance with requirements, an organization can reconcile information hitherto kept in different places. Business processes/workflows and business rules come to mind very quickly. In that respect, clearly, an ontology is much more than a data dictionary. It can also be the repository for process/workflow definitions and business rules.

Business processes and business rules are just two areas that benefit from a standardized information model. We see industry groups making efforts in this direction, the results of which can form essential parts of a business ontology.

The Business Process Modeling Notation (BPMN), which has been developed under the auspices of the Business Process Management Initiative (BPMI.org), has an information model at its core that can be represented in a business ontology. Version 1.0 of the BPMN specification was released in May 2004.
The Business Process Management Ontology (BPMO), developed by Jenz & Partner, is based on BPMN, and is the very first implementation of the specification with an ontology. The BPMO allows business analysts and domain experts to define business processes in a vendor-neutral format from which executable process definitions can be generated in some target process definition language.

Business rules were long given too little attention. Typically, business rules were hard coded into the application or “parameterized,” meaning that business rules would be kept outside applications in files in some proprietary format. Alternatively, representing a more modern approach, business rules can be formulated in some business rule language. A rule engine would interpret business rules at run time. However, current business rule languages are limited in scope and proprietary. None of these business rule languages is widely used on a commercial basis.

In 2003, the Object Management Group (OMG) issued a Business Semantics of Business Rules Request for Proposal (RFP). The Business Rules Team (BRT) provided an initial response to the RFP, which is work in progress. Eventually, the BRT work may well form the foundation for a standardized business rules language. A business rules ontology will be defined once the specification is sufficiently stable.

Standardization groups usually do not synchronize their work, resulting in domain specifications that are not well aligned with each other. Although it is certainly not justified to talk of conflicting specifications, more often than not, specifications overlap, and work groups use different terms for the same concepts.

A business ontology is the ideal place to reconcile specifications into a coherent and overarching information model. It can then be the source for the generation of various artifacts, such as deployable and executable business processes, business rules, and so on. The business ontology is expressed in a language that is both close to natural language and machine-interpretable.

The above figure shows the architecture of the Jenz & Partner Business Process Management Ontology (BPMO), which integrates multiple ontologies into a coherent whole. It has evolved into a full-blown generic business architecture over the past year, supporting reuse of generic concepts in different contexts. For example, roles and business rules are used in the business process management context as well as in industry-specific and organization-specific contexts. Hence, the roles and business rules ontologies are part of the generic business domain ontology layer.
Based on a sound information architecture, industry bodies and consulting firms can develop and make available pre-loaded repositories. An “out-of-the-box” repository would contain definitions for frequently used business processes and typical business rules on a per-industry basis. A user organization would be in the position to immediately generate deployable and executable business process definitions needed to run its core business operations.

**A Semantic Repository Backbone for the Masses**

At present, mostly large organizations make use of enterprise repositories. Enterprise repository software comes with a price tag, which is typically above the $100,000 mark. Hence, less than 2% of all organizations actually use enterprise repository software. To help with market penetration in the field of medium-sized and smaller organizations, enterprise repository software must become available at affordable license fees.

Even the use of an enterprise repository does not eliminate the need to cope with the plethora of independent tool repositories, which still exist in parallel. As mentioned above, tool vendors generally provide tool-specific repositories, which implement proprietary information models. Information exchange between the enterprise repository and multiple tool repositories must be routinely performed to keep enterprise repository and tool repositories in sync. The enterprise repository software exposes an API that can be used for importing information into the enterprise repository and exporting information from it. Import/export formats may be proprietary or compliant with some standardized data interchange format, such as XMI (XML Metadata Interchange), which has been defined by the Object Management Group (OMG).

The use of an ontology instead of enterprise repository software will not do away with the need to synchronize repositories. While the issues stay largely the same, the price tag is a different one. Although ontology management software is still in an early stage of the technology life-cycle, expectations are that the software will come fairly cheap, if not in the form of open source software. For example, the Protégé Ontology Editor, which incorporates some ontology management functionality, is popular open source software, with an installed base exceeding the 20,000 mark (as of September 2004).

If expectations materialize, ontology management software can be considered a semantic repository backbone “for the masses.” It will be affordable for all kinds of organizations. In particular, consulting firms will find opportunities to offer new types of services to their clients.

The semantic repository backbone will provide for unprecedented information integration, enabling users to perform queries on the integrated information base, which is impossible in an “information island” environment.

**Will Purpose-Built Tools Become Redundant?**

The question arises whether an ontology editor will make purpose-built tools, such as business process modeling tools, redundant. There are three major areas of functionality that a business process-modeling tool provides: graphical business process editing, business process validation, and generation of executable business process definitions.

An ontology editor provides only limited graphical editing capabilities. For example, Protégé, which is probably the most popular ontology editor today, has a graph widget, which does provide graphical editing capabilities, albeit in a somewhat primitive fashion. There are only a couple of basic graphical symbols, which is a limiting factor. It is not possible to represent the full power of the graphical BPMN notation. There are two options to get around this shortcoming: A business analyst may import a business process definition into a purpose-built business process modeling tool for viewing, or a transformation step may be necessary, which will produce a format that some other graphics tool can import and subsequently visualize the process flow. In both cases, however, the tool would do an automatic layout of the process flow, which is not necessarily adequate in all situations.

Business process validation can be performed in the ontology editor. An editor plug-in is required, which checks whether modeling elements have been used in a proper context, whether connections
between nodes are valid, and so on. Validation can only be performed when a process model is logically complete. Hence, validation is performed at the explicit request of the business analyst.

An ontology editor plug-in can generate executable business processes. By early October 2004, a Protégé plug-in will become available that lets a user generate business processes in XPDL. Generators for other business process definition languages, such as BPEL (Business Process Execution Language), will follow.

In all, an ontology editor still falls short with respect to the first area of functionality. This opens up an opportunity for open source projects, for example, to develop rich business process editing functionality. The result would be unprecedented integration of the semantic repository backbone and graphical editing functionality.

Thanks to the plug-in-friendly architecture of modern ontology editor tools, special-purpose plug-ins can be developed for editing business rules, organization charts, and so on. In the long term, this would allow for the phasing-out of current generation modeling tools that require their own repositories.

**Conclusion and Outlook**

Making effective use of an enterprise repository will become an option for medium-sized and even small enterprises, which often cannot afford the license fees and all the additional expenses incurred with the introduction of enterprise repository software. The ontology-based semantic repository backbone will become a reality.

Front-end tools will extend the semantic repository backbone to form the semantic business development control center. Ontology editors provide interfaces that facilitate development of plug-ins, which support business process modeling, business rule modeling, and so on.

The expectation is that generator plug-ins will become available for ontology editor tools, which can generate artifacts from the repository. For example, it would be possible to generate deployable business process definitions in multiple target process definition languages. Soon, it will be possible to generate executable process definitions in XPDL (XML Process Definition Language) from vendor neutral process definitions in the Business Process Management Ontology (BPMO).

Thanks to OWL, which is XML-based, it is possible to make use of XML-related tools and technologies, such as XSL transformation (XSLT), to create virtually any target format. This would give power users the freedom to create artifacts tailored to specific need, without having to rely on a limited selection of pre-defined reports.

In addition, an ontology language has the potential to form the “common language” spoken by domain experts and IT experts alike. Today, OWL is making its way from the academic domain into the software development mainstream. Although it is not uncommon to find ontologies in use today in large organizations and in IT consulting firms, it will take some time until OWL becomes a “lingua franca” like SQL, for example.

In all, the ontology-based approach represents a standards-based and strategic, rather than tactical, move towards the semantic business development control center. From a manager’s viewpoint, the implementation of the semantic business development control center is the effective application of a risk mitigation strategy. It reduces dependency on tools and tool vendors, who often add proprietary features, which may prove a bondage when a switch of tools is to be performed. As such, the semantic business development control center is a means to make switching tools much less painful.

The semantic business development control center is a key piece in an organization’s strategy to better align IT with business. It will help business and IT experts speak the same language.

-----

**Dieter Jenz** is the President of Jenz & Partner GmbH. He can be reached at info@jenzundpartner.de
**Jenz & Partner** was founded in 1985. Dieter E. Jenz serves as the company's president. The company provides a range of industry analyst, business and technical consulting, and educational services. It is widely known for its contributions to distributed applications, object-oriented development, and relational database technology. For additional information, Jenz & Partners' Website URL is [www.jenzundpartner.de](http://www.jenzundpartner.de).

Jenz & Partner has developed a business process ontology, which can be made available to clients in the context of consulting engagements. Jenz & Partner provides training in ontology definition and in software development process optimization in general.

Although this information is believed to have been accurate at the time of publication, Jenz & Partner GmbH cannot and does not warrant the accuracy, completeness or suitability of this information or that the information is correct currently. Jenz & Partner GmbH and/or Jenz & Partner GmbH analysts and consultants cannot be held liable for any damages caused by or alleged to be caused directly or indirectly by decisions made using any Jenz & Partner GmbH material. This publication and features described herein are subject to change without notice.

Names appearing in this document that are registered trademarks are not mentioned as being so, nor is the trademark symbol inserted with each mention of these registered trademarks. Product names and company names mentioned in this publication are the property of their respective owners. In no way does Jenz & Partner GmbH have the intention of infringing on any registered trademark.

Copyright © 2004 by Jenz & Partner GmbH. All rights reserved. No part of this publication may be reproduced by any method whatsoever, without the prior written consent of Jenz & Partner GmbH.