

Driving a Coherent IT Infrastructure with an Enterprise Ontology

WHITEPAPER:

DATA INTEGRATION, QUALITY, AND MANAGEMENT



Todd Morley • Product Architect, Pitney Bowes Business Insight

Driving a Coherent IT Infrastructure with an Enterprise Ontology

2

ABSTRACT

IT ORGANIZATIONS TODAY FACE DATA LANDSCAPES THAT OFTEN CONTAIN MULTIPLE, INCONSISTENT REPRESENTATIONS OF THE SAME KIND OF DATA. THIS HETEROGENEITY COMPLICATES DATA-INTEGRATION ACTIVITIES SUCH AS BUILDING THE ETL PROCESS FOR AN ENTERPRISE DATA WAREHOUSE. IT ALSO REFLECTS AND LEADS TO UNNECESSARY REPETITION OF DATA-DESIGN DECISIONS. AN ENTERPRISE ONTOLOGY CATALOGS ALL REPRESENTATIONS OF A GIVEN KIND OF DATA, SPECIFIES WHICH OF THEM ARE PRIMARY, AND DEFINES A CANONICAL REPRESENTATION FOR THEM. IT CAN USE THE ONTOLOGY TO IMPROVE REPRESENTATIONAL CONSISTENCY, ENSURE INCLUSIVENESS AND ACCURACY IN A DATA WAREHOUSE, AVOID UNNEEDED DATA DESIGN, AND REDUCE DESIGN AND DEVELOPMENT COSTS.

ENTERPRISE ONTOLOGIES CAN BE EXTREMELY USEFUL. WHEN PUBLISHED TO A USER COMMUNITY, AN ENTERPRISE ONTOLOGY BECOMES AN INTELLIGENT-SEARCH TOOL (ESPECIALLY IF THE ONTOLOGY IS WRITTEN IN A LANGUAGE SUPPORTING INFERENCE).

Background

The word ‘ontology’ has a colorful history. Until the 20th century, ‘ontology’ was the branch of philosophy that addressed the questions “What exists?” and “What is the nature of existence?” Twentieth-century analytic philosophers such as Quine, Carnap, and Goodman disparaged such general questions as metaphysical, meaningless. They styled themselves “ontological relativists” and spoke of one’s “ontological commitments.” For them an ontology was a universe of discourse, a collection of terms one is willing to use in discussing a given topic, together with interpretations of these terms, and rules governing their use: jargon. The analytic philosophers’ ideal was to excise from philosophical discourse the twin evils of ambiguity and vagueness—to have exactly one word for one kind of thing, and always to be able to decide whether a given thing was of a given kind.

Computer science soon adopted the idea of an ontology. The idea’s popularity continues to grow. In 2004 W3C published its Web Ontology Language (OWL) Recommendation. In the same year the ISO and IEC published their standard governing ontologies (*ISO/IEC 11179: Information Technology—Metadata Registries*). A decade of active academic research has produced a number of freely available ontology editors. TopQuadrant’s TopBraid is perhaps the first commercial editor; it became available in 2006. The same year saw a surge of texts on the topic, including the first specifically about enterprise ontologies (Dietz 2006). So it should come as no surprise that forward-thinking IT organizations have only recently started creating enterprise ontologies, documents that describe and/or prescribe an IT organization’s universe of discourse.

Enterprise ontologies can be extremely useful. When published to a user community, an enterprise ontology becomes an intelligent-search tool (especially if the ontology is written in a language supporting inference). Web visionaries see ontologies supporting extreme forms of distributed and autonomous search (see the W3C recommendation’s use cases for details). This paper focuses on how an enterprise

ontology is useful to the IT organization itself, particularly when building and supporting a proprietary software infrastructure.

What is an Enterprise Ontology?

Content. An enterprise ontology resembles a dictionary or encyclopedia. (In the database-architecture realm, an ontology may be captured as an “enterprise data dictionary” or “logical model” using tools such as Embarcadero Technologies’ ER/Studio®.) Each entry describes an entity class modeled by one or more of the IT organization’s software systems. In this respect an enterprise ontology is quite like an object-oriented class hierarchy—the most obvious difference being that an enterprise ontology exists independent of any programming language or paradigm.

In spite of the ISO/IEC standard, enterprise ontologies take many forms. Here is a short list of items that are likely to appear in any enterprise-ontology entry:

- The **canonical identifier** is the entity class’ primary name.
- The **definition** or **description** explains what the entity class models.
- **Language- and paradigm-specific identifiers** are secondary names that are, or should be, used to represent instances of the entity class in particular programming languages or paradigms.
- **Atomic entity classes** have just one data element, and so have a single **data type**. (Some approaches treat atomic entity classes as “elements.”)
- **Non-atomic entity classes** have multiple **components**, each of which is itself an entity class.
- Entity classes can have **logical relations** with other entity classes (including inheritance); and their components can have logical relations with each other, and with the entity class itself. (These relations can have **cardinalities**, as they do in database modeling.)

Driving a Coherent IT Infrastructure with an Enterprise Ontology

4

- The organization's systems can implement a variety of **operations** on an entity class. (One could model an operation as a degenerate entity class, that is, an entity class having no components.)
- The IT organization's systems may have several **representations** of the entity class. Each representation's data set can have a **data profile**, summary statistics describing the data set's size and diversity.

Here is a sample atomic entity class, presented informally (as it might appear in a static document or an unstructured intranet Wiki):

Last Name

- A **last name** is a person's surname or family name. It can be named `LAST_NAME` or `LastName`, and is a string having at most 50 characters. A last name is a component of a **full name**. The **name spell checker** operation validates **last name** spellings. The `HR.EMPLOYEE.LAST_NAME` and `SALES.CUSTOMER.LAST_NAME` database columns are the primary sources of employee and customer **last names**, respectively. The former column contains about 12,000 distinct values in about 25,000 entries; the latter column contains about 2,800 distinct values among 5,500 entries. The Java Employee and Customer classes both contain several representations of **last name**.

And here is a sample non-atomic entity class:

Full Name

- A **full name** is a person's entire name. It can be named `FULL_NAME` or `FullName`. It consists of one **first name**, an optionally null **middle name**, and one **last name**; serialized, it becomes a string having at most 152 characters (including spaces between the components). The **name spell checker** operation validates **full name** spellings. The `HR.EMPLOYEE.FULL_NAME` and `SALES.CUSTOMER.FULL_NAME` database columns

represent **full name**; both are denormalized copies of their components' primary representations (in the same tables). Most of the 25,000 employee full names and 5,500 customer full names are unique. The Java Employee and Customer classes both contain several representations of **full name**.

In both examples, each occurrence of a canonical name appears in bold. (If the entries were in a wiki, the canonical names would be hyperlinks to their entity-class entries.) The secondary names follow conventions for database and object-oriented programming, respectively. The data type of the atomic entity class is language independent, and presumably has standard representations in all of the programming languages that the enterprise uses. The list of the non-atomic entity class' components mentions the cardinality of the relations between the entity class and its components. The database representations are flagged as primary sources or denormalized representations (other database representations could be foreign keys, rather than actual instances).

Descriptive vs. prescriptive ontologies. An enterprise ontology can be both descriptive and prescriptive. Descriptive ontological data are retrospective; prescriptive data are prospective. When both kinds of data appear in an ontology, it becomes a powerful tool for rationalizing IT infrastructure. (More on this below.)

Repositories. There are many ways to record and publish an enterprise ontology. It can be a static HTML, Acrobat®, or Word® document; a wiki; or a formal model constructed in an ontology editor or database-modeling tool. Some of these forms are better suited to centralized ontology-development and -maintenance processes, while others lend themselves well to decentralized processes. This distinction can have a dramatic effect on the enterprise ontology's place in the organization's workflows, so management should attend to it carefully.

AN ORGANIZATION WITH A MODEST DATA LANDSCAPE CAN GIVE A SINGLE EMPLOYEE (AN “ENTERPRISE DATA ARCHITECT” OR “ENTERPRISE INFORMATION ARCHITECT”) RESPONSIBILITY FOR PRODUCING AND MAINTAINING THE ENTERPRISE ONTOLOGY.

What Role can an Enterprise Ontology play in IT Development?

Descriptive ontologies. A descriptive ontology has many uses. Enterprise architects can use it to identify unnecessarily redundant or inconsistent representations, poorly structured data flows, and similar opportunities to improve enterprise architecture. For example, an enterprise architect writing a prescriptive entry for a given entity class would capture in the prescriptive model all of the components identified in descriptions of pre-existing representations of the entity class. A data-warehouse architect could then rely on the prescriptive entry to model an enterprise-data-warehouse dimension table for the entity class, confident that the dimension would capture all available data for that class. Likewise, application architects can consult a descriptive ontology to find the primary data source for an ETL process or a data service. Application programmers can use a descriptive ontology to interpret and maintain existing code. New hires and consultants can read a descriptive ontology to acquaint themselves quickly with the organization’s code base.

Prescriptive ontologies. A prescriptive ontology captures design decisions at several levels of abstraction. The development process must be altered to make sure these decisions are made, when they have not already been made, at the beginning of a development cycle (such as an Agile iteration), and immediately documented in the ontology. Thereafter, developers should consult the ontology when

- **designing a service, class, or function interface.** The design’s structure should instantiate the entity class’ structure. The names of variables, arguments, inputs, and outputs should be chosen from the entity class’ language- or paradigm-specific names.
- **designing a database schema.** The schema’s structure should reflect that of the entity class and its relations with other entities. Table and column names should be chosen from the entity class’ language- or paradigm-specific names.

- **designing an ETL process.** ETL processes should retrieve data from primary sources whenever possible, and should transform the data to conform with the entity class’ structure.

Ontology-based workflows. An organization with a modest data landscape can give a single employee (an “enterprise data architect” or “enterprise information architect”) responsibility for producing and maintaining the enterprise ontology. Here the architect must become a subject-matter expert (SME) for each data source with which the architect is not already familiar, to develop descriptions of the data source’s entities. Development teams submit data-design requirements to the architect, who makes and documents all data-design decisions. (The architect may even conduct code reviews to enforce fine-grained conformance to e.g. nomenclature decisions.) Single-user ontology editors and static documents are satisfactory repositories for ontologies produced by this sort of highly centralized workflow.

Organizations having more complex data landscapes are unlikely to produce a complete enterprise ontology, or indeed integrate it into the software-development process, through a fully centralized workflow. The enterprise data architect becomes overwhelmed by the sheer quantity of retrospective modeling to be done, or by the number of development projects requiring prospective modeling, and so becomes a development-process bottleneck. In such cases it is necessary to delegate the initial draft of an entity class’ ontology entry to SMEs such as business analysts, technical analysts, application architects, and senior developers, according to their areas of expertise. The enterprise data architect revises the drafts as necessary to achieve completeness and consistency with sound design principles, nomenclature conventions, etc., and publishes reviewed entries in the ontology. Here multi-user collaboration tools that support versioning or review processes (at the cost of not implementing an ontology in OWL) become attractive ontology repositories. Structured-wiki tools such as TWiki® become especially attractive. (The author predicts that commercial ontology editors will eventually become structured wikis that support ontology languages such as OWL.) Peer-review processes also become valuable aids to ensure that code and schema designs conform with the enterprise ontology.

Driving a Coherent IT Infrastructure with an Enterprise Ontology

6

Why does my Organization Want an Enterprise Ontology?

An enterprise ontology creates a high level of architectural coherence and transparency, as well as encouraging rapid development, in an IT organization's custom applications. Because the ontology largely determines how an entity class is modeled in databases, services, and object-oriented class hierarchies, design decisions are made only once, recorded in the ontology, then re-used as often as necessary—in all possible contexts (not just one programming language or paradigm). This reduces the demand for architectural activity, while freeing developers to devote more of their attention to correct, efficient implementation. Code becomes self-documenting and easy to maintain, especially if the ontology is careful to enforce plain-language nomenclature conventions, because coders no longer create idiosyncratic or cryptic nomenclature. Unnecessary transformations are avoided when developing ETL.

Some large IT organizations have at least as much to gain from a descriptive ontology as from its prescriptive counterpart. This is especially true where the organization has spent years producing poorly documented “one-off” applications, operational datastores, data marts, and services without a strong enterprise-architecture function. In such cases, technical-employee turnover means that eventually the data landscape is all but unintelligible to most of the organization. Writing a descriptive ontology then becomes a collective effort to reverse engineer the data landscape, a first step towards integrating it into (say) an enterprise service bus and an enterprise data warehouse. Once the descriptive ontology exists, the enterprise-architecture function can analyze it to identify opportunities to consolidate, reconcile, and streamline components of the enterprise's software infrastructure; and then management can prioritize these opportunities, and the real work can begin.

REFERENCES

- Albanil, Antonia, and Jan L. G. Dietz (2005). “Identifying Business Components on the Basis of an Enterprise Ontology.” <http://interop-esa05.unige.ch/INTEROP/Proceedings/Interop-ESAScientific/PerPaper/110-3%20358.pdf>.
- Carnap, Rudolf (1988). *Meaning and Necessity: A Study in Semantics and Modal Logic*, Midway Reprint edition. Chicago: The University of Chicago Press.
- Dietz, Jan L.G. (2006). *Enterprise Ontology: Theory and Methodology*. Germany: Springer-Verlag.
- Embarcadero Technologies (2005). *ER/Studio® Quick-Start Guide*. <http://www.embarcadero.com/>.
- Goodman, Nelson (1977). *The Structure of Appearance*. Boston: D. Reidel Publishing.
- International Standards Organization (ISO) and International Electromechanical Commission (IEC) (2004). *ISO/IEC 11179: Information Technology—Metadata Registries*. <http://metadata-standards.org/11179/>.
- McComb, Dave (2006). “The Enterprise Ontology.” <http://www.tdan.com/i037fe03.htm>.
- Quine, W.V. (1976) *The Ways of Paradox and Other Essays*, Revised and Enlarged edition. Cambridge: Harvard University Press.
- Schreiber, Zvi (2003). “Applying the Semantic Web Vision to Enterprise Data Management: A Case Study.” <http://www2003.org/cdrom/papers/alternate/P079/p79-schreiber-WWW2003.htm>.
- TopQuadrant Inc. (2006). *TopBraid Composer Getting Started Guide Version 1.0*. <http://www.topbraidcomposer.com/docs/TBC-Getting-Started-Guide.pdf>.
- Thoeny, Peter (2007). “TWiki”—Enterprise Wiki and Collaboration Platform.” <http://twiki.org/>.
- W3C (2004). “OWL Web Ontology Language Overview.” <http://www.w3.org/TR/owl-features/>.



UNITED STATES

One Global View
Troy, NY 12180-8399

main: 518.285.6000
1.800.327.8627
fax: 518.285.6070

pbbi.sales@pb.com
www.pbinsight.com

CANADA

26 Wellington Street East
Suite 500
Toronto, Ontario
M5E 1S2

main: 416.594.5200
fax: 416.594.5201

pbbi.canada.sales@pb.com
www.mapinfo.ca