A New Methodology for Building Lightweight, Domain Ontologies

by Mike Bergman - Wednesday, September 01, 2010

http://www.mkbergman.com/908/a-new-methodology-for-building-lightweight-domain-ontologies/

Bringing Ontology Development and Maintenance to the Mainstream

Ontologies supply the structure for relating information to other information in the semantic Web or the linked data realm. Ontologies provide a similar role for the organization of data that is provided by relational data schema. Because of this structural role, ontologies are pivotal to the coherence and interoperability of interconnected data [1]. There are many ways to categorize ontologies. One dimension is between upper level and mid- and lower- (or domain-) level. Another is between reference or subject (domain) ontologies. Upper-level ontologies [2] tend to be encompassing, abstract and inclusive ways to split or organize all "things". Reference ontologies tend to be cross-cutting such as ones that describe people and their interests (*e.g.*, FOAF), reference subject concepts (*e.g.*, UMBEL), bibliographies and citations (*e.g.*, BIBO), projects (*e.g.*, DOAP), simple knowledge structures (*e.g.*, SKOS), social networks and activities (*e.g.*, SIOC), and so forth. The focus here is on domain ontologies, which are descriptions of particular subject or domain areas. Domain ontologies are the "world views" by which organizations, communities or enterprises describe the concepts in their domain, the relationships between those concepts, and the instances or individuals that are the actual things that populate that structure. Thus, domain ontologies are the basic bread-and-butter descriptive structures for real-world applications of ontologies. According to Corcho *et al.* [3] "a domain ontology can be extracted from special purpose encyclopedias, dictionaries, nomenclatures, taxonomies, handbooks, scientific special languages (say, chemical formulas), specialized KBs, and from experts." Another way of stating this is to say that a domain ontology -- properly constructed -- should also be a faithful representation of the language and relationships for those who interact with that domain. The form of the interaction can range from work to play to intellectual understanding or knowledge.

"... *ontology engineering research should strive for a unified, lightweight and*

component-based methodological framework, principally targeted at domain experts ...."
Simperl *et al.* [4]

Another focus here is on lightweight ontologies. These are typically defined as more hierarchical or classificatory in nature. Like their better-known cousins of *taxonomies*, but with greater connectedness, lightweight ontologies are often designed to represent subsumption or other relationships between concepts. They have not too many or not too complicated predicates (relationships). As relationships are added and the complexities of the world get further captured, ontologies migrate from the lightweight to the "heavyweight" end of the spectrum. The development of ontologies goes by the names of ontology engineering or *ontology building*, and can also be investigated under the rubric of ontology learning. For reasons as stated below, we prefer not to use the term *ontology engineering*, since it tends to convey a priesthood or specialized expertise in order to define or use them. As indicated, we see ontologies as being (largely) developed and maintained by the users or practitioners within a given domain. The tools and methodologies to be employed need to be geared to these same democratic (small "d") objectives.

A Review of Prior Methodologies

For the last twenty years there have been many methods put forward for how to develop ontologies. These methodological activities have diminished somewhat in recent years. Yet the research as separately discussed in Ontology Development Methodologies [1] seems to indicate this state of methodology development in the field:

- Very few uniquely different methods exist, and those that do are relatively older in nature
- The methods tend to either cluster into incremental, iterative ones or those more oriented to comprehensive approaches
- There is a general logical sharing of steps across most methodologies from assessment to deployment and testing and refinement
- Actual specifics and flowcharts are quite limited; with the exception of the UML-based systems, most appear not to meet enterprise standards
- The supporting toolsets are not discussed much, and most of the examples if at all are based solely on a single or governing tool. Tool integration and interoperability is almost non-existent in terms of the narratives, and
- Development methodologies do not appear to be an active area of recent research.

While there is by no means unanimity in this community, some general consenses can be seen from these prior reviews, especially those that concentrate on practical or enterprise ontologies. In terms of design objectives, this general consensus suggests that ontologies should be [4]:

- Collaborative
- Lightweight
- Domain-oriented (subject matter and expertise)
- Integrated, and

- Incremental.

While laudable, and which represent design objectives to which we adhere, current ontology development methods do not meet these criteria. Furthermore, to be discussed in our next installment, there is also an inadequate slate of tools ready to support these objectives.

A Call for a New Methodology

If you ask most knowledgeable enterprise IT executives what they understand ontologies to mean and how they are to be built, you would likely hear that ontologies are expensive, complicated and difficult to build. Reactions such as these (and not trying to set up strawmen) are a reflection of both the lack of methods to achieve the consensual objectives above and the lack of tools to do so. The use of ontology design patterns is one helpful approach [5]. Such patterns help indicate best design practice for particular use cases and relationship patterns. However, while such patterns should be part of a general methodology, they do not themselves constitute a methodology. Also, as Structured Dynamics has argued for some time, the future of the semantic enterprise resides in ***ontology-driven apps*** [6]. Yet, for that vision to be realized, clearly both methods and tools to build ontologies must improve. In part this series is a reflection of our commitment to plug these gaps. What we see at present for ontology development is a highly technical, overly engineered environment. Methodologies are only sparsely or generally documented. They are not lightweight nor collaborative nor really incremental. While many tools exist, they do not interoperate and are pitched mostly at the professional ontologist, not the domain user. In order to achieve the vision of ontology-driven apps the methods to develop the fulcrum of that vision -- namely, the ontologies themselves -- need much additional attention. An adaptive methodology for ontology development is well past due.

Design Criteria for an Adaptive Methodology

We can thus combine the results of prior surveys and recommendations with our own unique approach to adaptive ontologies in order to derive design criteria. We believe this adaptive approach should be:

- Lightweight and domain-oriented
- Contextual
- Coherent
- Incremental
- Re-use structure
- Separate the ABox and TBox (separate work), and
- Simpler, with interoperable tools designs.

We discuss each of these design criteria below. While we agree with the advisability of collaboration as a design condition -- and therefore also believe that tools to support this methodology must also accommodate group involvement -- collaboration *per se* is not a design requirement. It is an implementation best practice. Effective ontology development is as much as anything a matter of mindset. This mindset is grounded in leveraging what already exists, "paying as one benefits" through an incremental approach, and starting simple and adding complexity as understanding and experience are gained. Inherently this approach requires domain users to be the driving force in ongoing development with appropriate tools to support that emphasis. Ontologists and ontology engineering are important backstops, but not in the lead design or development roles. The net result of this mindset is to develop pragmatic ontologies that are understood -- and used by -- actual domain practitioners.

Lightweight and Domain-oriented

By definition the methodology should be lightweight and oriented to particular domains. Ontologies built for the pragmatic purposes of setting context and aiding interoperability tend to be lightweight with only a few predicates, such as `isAbout`, `narrowerThan` or `broaderThan`. But, if done properly, these lighter weight ontologies can be surprisingly powerful in discovering connections and relationships. Moreover, they are a logical and doable intermediate step on the path to more demanding semantic analysis.

Contextual

Context simply means there is a reference structure for guiding the assignment of what content 'is about' [7]. An ontology with proper context has a balanced and complete scope of the domain at hand. It generally uses fairly simple predicates; Structured Dynamics tends to use the UMBEL vocabulary for its predicates and class definitions, and to link to existing UMBEL concepts to help ensure interoperability [8]. A good gauge for whether the context is adequate is whether there are sufficient concept definitions to disambiguate common concepts in the domain.

Coherent

The essence of ***coherence*** is that it is a state of consistent connections, a logical framework for integrating diverse elements in an intelligent way. So while ***context*** supplies a reference structure, ***coherence*** means that the structure makes sense. With relation to a content graph, this means that the right connections (edges or predicates) have been drawn between the object nodes (or content) in the graph [9]. Relating content coherently itself demands a coherent framework. At the upper reference layer this begins with UMBEL, which itself is an extraction from the vetted and coherent Cyc common sense knowledge base. However, as domain specifics get added, these details, too, must be testable against a unified framework. Logic and coherence testing are thus an essential part of the ontology development methodology.

Incremental

Much value can be realized by starting small, being simple, and emphasizing the pragmatic. It is OK to make those connections that are doable and defensible today, while delaying until later the full scope of semantic complexities associated with complete data alignment. An open world approach [10] provides the logical basis for incremental growth and adoption of ontologies. This is also in keeping with the continuous and incremental deployment model that Structured Dynamics has adopted from MIKE2.0 [11] . When this model is applied to the process of ontology development, the basic implementation increments appear as follows:

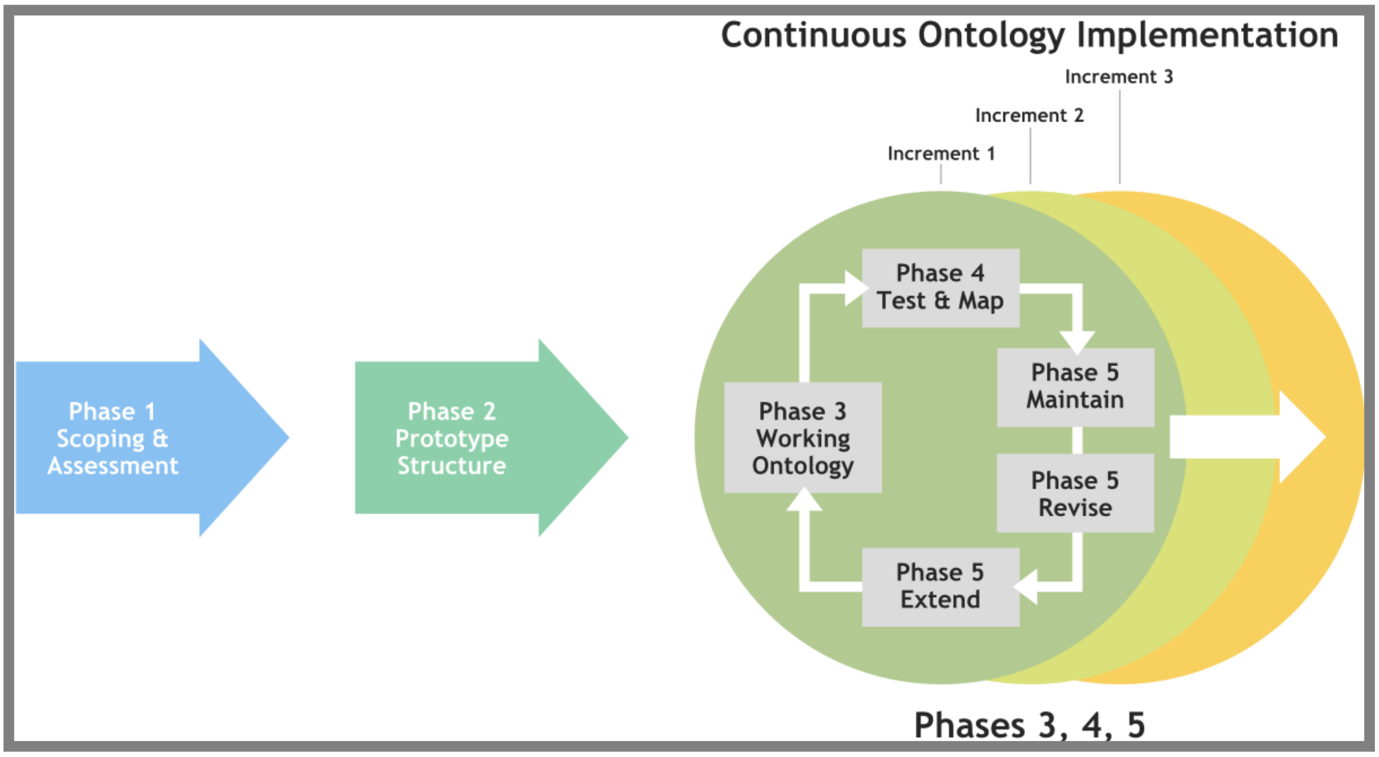

Figure 1. A Phased, Incremental Approach to Ontology Development *(click to expand)*

The first two phases are devoted to scoping and prototyping. Then, the remaining phases of creating a working ontology, testing it, maintaining it, and then revising and extending it are repeated over multiple increments. In this manner the deployment proceeds incrementally and only as learning occurs. Importantly, too, this approach also means that complexity, sophistication and scope only grows consistent with demonstrable benefits.

Re-use of Structure

Fundamental to the whole concept of coherence is the fact that domain experts and practitioners have been looking at the questions of relationships, structure, language and meaning for decades. Though perhaps today we now finally have a broad useful data and logic model in RDF, the fact remains that massive time and effort has already been expended to codify some of these understandings in various ways and at various levels of completeness and scope. These are prior investments in structure that would be silly to ignore. Yet, today, most methodologies do ignore these resources. This ignorance of prior investments in information relationships is perplexing. Though unquestioned adoption of legacy structure is inappropriate to modern interoperable systems, that fact is no excuse for re-inventing prior effort and discoveries, many of which are the result of laborious consensus building or negotiations. The most productive methodologies for modern ontology building are therefore those that re-use and reconcile prior investments in structural knowledge, not ignore them. These existing assets take the form of already proven external ontologies and internal and industry structures and vocabularies.

Separation of the ABox and TBox

Nearly a year ago we undertook a major series on ***description logics*** [12], a key underpinning to Structured Dynamics' conceptual and logic foundation to its ontology development. While we can not always adhere to strict and conforming description logics designs, our four-part series helped provide guidance for the separation of concerns and work that can also lead to more effective ontology designs [13]. Conscious separation of the so-called ABox (assertions or instance records) and TBox (conceptual structure) in ontology design provides some compelling benefits:

- Easier ingest and incorporation of external instance data, including conversion from multiple formats and serializations
- Faster and more efficient inferencing and analysis and use of the conceptual structure (TBox)
- Easier federation and incorporation of distributed data stores (instance records), and
- Better segregation of specialized work to the ABox, TBox and specialty work modules, as this figure shows [14]:

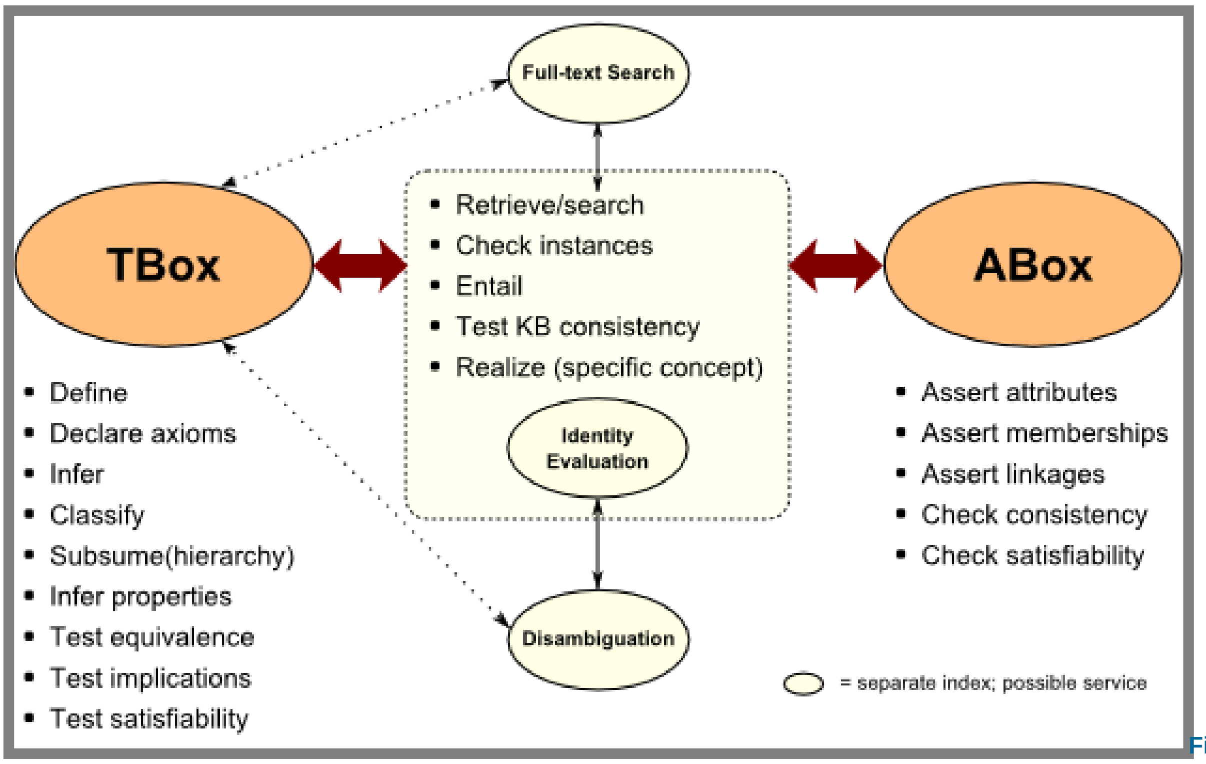

Figure 2. Separation of the TBox and ABox [14]

Maintaining identity relations and disambiguation as separate components also has the advantage of enabling different methodologies or algorithms to be determined or swapped out as better methods become available. A low-fidelity service, for example, could be applied for quick or free uses, with more rigorous methods reserved for paid or batch mode analysis. Similarly, maintaining full-text search as a separate component means that work can be done by optimized search engines with built-in faceting.

Simple, Interoperable Tools Support

An essential design criteria is to have a methodology and work flow that explicitly accounts for simple and interoperable tools. By "simple" we mean targeted, task-specific tools and functionality that is also geared to domain users and practitioners. Of all design areas, this one is perhaps the weakest in terms of current offerings. The next installment in this series [1] will address this topic directly.

The New Methodology

Armed with these criteria, we are now ready to present the new methodology. In summary terms, we can describe the steps in the methodology as:

1. Scope, analyze, then leverage existing assets
2. Prototype structure
3. Pivot on the working ontology
4. Test
5. Use and maintain
6. Extend working ontology and repeat.

Two Parallel Tracks

After the scoping and analysis phase, the effort is split into two tracks:

- Instances, and their descriptive characteristics, and
- Conceptual relationships, or ontologies.

This split conforms to the separation of ABox and TBox noted above [15]. There are conceptual and workflow parallels between entities and data *v*. ontologies. However, the specific methodologies differ, and we only focus on the conceptual ontology side in the discussion below, shown as the upper part (blue) of Figure 3:

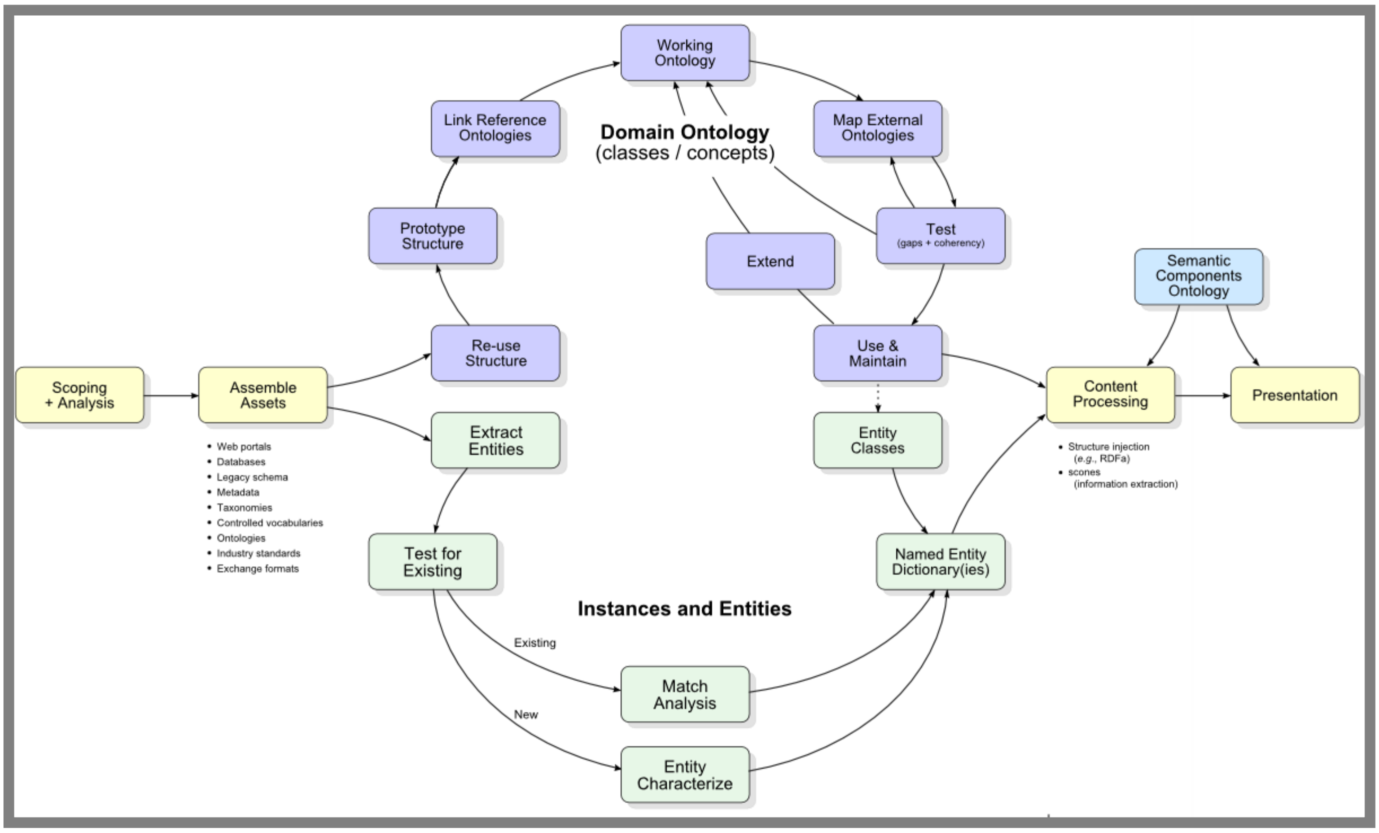

Figure 3. Flowchart of Ontology Development Methodology [16] *(click to expand)*

Two key aspects of the initial effort are to properly scope the size and purpose of the starting prototype and to inventory the existing assets (structure and data; internal and external) available to the project.

Re-Use Structure

Most current ontology methodologies do not emphasize re-use of existing structure. Yet these resources are rich in content and meaning, and often represent years to decades of effort and expenditure in creation, assembly and consensus. Just a short list of these potential sources demonstrates the treasure trove of structure and vocabularies available for re-use: Web portals; databases; legacy schema; metadata; taxonomies; controlled vocabularies; ontologies; master data catalogs; industry standards; exchange formats, etc. Metadata and available structure may have value no matter where or how it exists, and a fundamental aspect of the build methodology is to bring such candidate structure into a common tools environment for inspection and testing. Besides assembling and reviewing existing sources, those selected for re-use must be migrated and converted to proper ontological form (OWL in the case of those developed by Structured Dynamics). Some of these techniques have been demonstrated for prior patterns and schema [17]; in other instances various converters, RDFizers or scripts may need to be employed to effect the migration. Many tools and options exist at this stage, even though as a formal step this conversion is often neglected.

Prototype Structure

The prototype structure is the first operating instance of the ontology. The creation of this initial structure follows quite closely the approach recommended in *Ontology Development 101* [18], with some modifications to reflect current terminology:

1. Determine the domain and scope of the ontology
2. Consider reusing existing ontologies
3. Enumerate important terms in the ontology
4. Define the classes and the class hierarchy
5. Define the properties of classes
6. Create instances

The prototype structure is important since it communicates to the project sponsors the scope and basic operation of the starting structure. This stage often represents a decision point for proceeding; it may also trigger the next budgeting phase.

Link Reference Ontologies

An essential aspect of a build methodology is to re-use "standard" ontologies as much as possible. Core ontologies are Dublin Core, DC Terms, Event, FOAF, GeoNames, SKOS, Timeline, and UMBEL. These core ontologies have been chosen because of universality, quality, community support and other factors [19]. Though less universal, there are also a number of secondary ontologies, namely BIBO, DOAP, and SIOC that may fit within the current scope. These are then supplemented with quality domain-specific ontologies, if such exist. Only then are new name spaces assigned for any newly generated ontology(ies).

Working Ontology

The working ontology is the first production-grade (deployable) version of the ontology. It conforms to all of the ontology building best practices and needs to be complete enough such that it can be loaded and managed in a fully conforming ontology editor or IDE [20]. By also using the OWL API, this working structure can also be the source for specialty tools and user maintenance functions, short of requiring a full-blown OWL editor. Many of these aspects are some of the poorest represented in the current tools inventory; we return to this topic in the next installment. The working ontology is the complete, canonical form of the domain ontology(ies) [21]. These are the central structures that are the focus for ongoing maintenance and extension efforts over the ensuing phases. As such, the ontologies need to be managed by a version control system with comprehensive ontology and vocabulary management support and tools.

Testing and Mapping

As new ontologies are generated, they should be tested for coherence against various reasoning, inference and other natural language processing tools. Gap testing is also used to discover key holes or missing links within the resulting ontology graph structure. Coherence testing may result in discovering missing or incorrect axioms. Gap testing helps identify internal graph nodes needed to establish the integrity or connectivity of the concept graph. Though used for different purposes, mapping and alignment tools may also work to identify logical and other inconsistencies in definitions or labels within the graph structure. Mapping and alignment is also important in its own right in order to establish the links that help promote ontology and information interoperability. External knowledge bases can also play essential roles in testing and mapping. Two prominent knowledge base examples are Cyc and Wikipedia, but many additional exist for any specific domain.

Use and Maintenance

Of course, the whole purpose of the development methodology is to create practical, working ontologies. Such uses include search, discovery, information federation, data interoperability, analysis and reasoning, The general purposes to which ontologies may be put are described in the Executive Intro to Ontologies [22]. However, it is also in day-to-day use of the ontology that many enhancements and improvements may be discovered. Examples include improved definitions of concepts; expansions of synonyms, aliases and jargon for concepts; better, more intuitive preferred labels; better means to disambiguate between competing meanings; missing connections or excessive connections; and splitting or consolidating of the underlying structure. Today, such maintenance enhancements are most often ***not*** pursued because existing tools do not support such actions. Reliance on IDEs and tools geared to ontology engineering are not well suited to users and practitioners being able to note or effect such changes. Yet ongoing ontology use and adaptation clearly suggest that users should be encouraged to do so. They are the ones in the front lines of identifying and potentially recording such improvements.

Extend

Ontology development is a process, not a static destination or event. This observation makes intuitive sense since we understand ontologies to be a means to capture our understanding of our domains, which is itself constantly changing due to new observations and insights. This factor alone suggests that ontology development methodologies must therefore give explicit attention to extension. But there is another reason for this attention. Incremental, adaptive ontologies are also explicitly designed to expand their scope and coverage, bite by bite as benefits prove themselves and justify that expansion. A start small and expand strategy is of course lower risk and more affordable. But, for it to be effective, it also must be designed explicitly for extension and expansion. Ontology growth thus occurs both from learning and discovery and from expanding scope. Versioning, version control and documentation (see below) thus assume more central importance than a more static view would suggest. The use of feedbacks and the continuous improvement design based on MIKE2.0 are therefore also central tenets of our ontology development methodology.

Documentation

This perspective of the ontology as a way to capture the structure and relationships of a domain -- which is also constantly changing and growing -- carries over to the need to document the institutional memory and use of it. Both better tools -- such as vocabulary management and versioning -- and better work processes need to be instituted to properly capture and record use and applications of ontologies. Some of these aspects are now handled with utilities such as OWLdoc or the TechWiki that Structured Dynamics has innovated to capture ontology knowledge bases on an ongoing basis. But these are still rudimentary steps that need to be enforced with management commitment and oversight. One need merely begin to probe the ontology development literature to observe how sparse the pickings are. Very little information on methodologies, best practices, use cases, recipes, how to manuals, conversion and use steps and other documentation really exists at present. It is unfortunately the case that documentation even lags the inadequate state of tools development in the ontology space.

Content Processing

Once formalized, these constructs -- the structured ontologies or the named entity dictionaries as shown in Figure 3 -- are then used for processing input content. That processing can range from conversion to direct information extraction. Once extracted, the structure may be injected (via RDFa or other means) back into raw Web pages. The concepts and entities that occur within these structures help inform various tagging systems [23]. The information can also be converted and exported in various forms for direct use or for incorporation in third-party systems. Visualization systems and specialized widgets (see next) can be driven by the structure and results sets obtained from querying the ontology structure and retrieving its related instance data. While these purposes are somewhat beyond the direct needs of the ontology development methodology, the ontology structures themselves must be designed to support these functions.

Semantic Component Ontology

In our methodology we also provide for administrative ontologies whose purpose is to relate structural understandings of the underlying data and data types with applicable end-use and visualization tools ("widgets"). Thus the structural knowledge of the domain gets combined with an understanding of data types and what kinds of visualization or presentation widgets might be invoked. The phrase ***ontology-driven apps*** results from this design. Amongst other utility ontologies, Structured Dynamics names its major tool-driver ontology the SCO (Semantic Component Ontology). The SCO works in intimate tandem with the domain ontologies, but is constructed and designed with quite different purposes. A description of the build methodology for the SCO (or its other complementary utility ontologies) is beyond the scope of this current document.

Tooling and Best Practices

As sprinkled throughout the above commentary, this methodology is also intimately related to tools and best practices. The next chapter in this series is devoted to and will be archived on the TechWiki as the lightweight domain ontology methodology. Best practices will be handled in a similar way for the chapter after that one and in its ontology best practices document on the TechWiki.

Time for a Leap Forward in Methodology

Earlier reviews and the information in this document suggest a real need for ontology building methodologies that are integrated, easier to use, interoperate with a richer tools set and are geared to practitioners versus priests. The good news is that there are architectures and building blocks to achieve this vision. The bad news is that the first steps on this path are only now beginning. The next two installments in this series add further detail for why it is time -- and how -- we can make a leap forward in methodology. Those critical remaining pieces are in tools and best practices.

---

[1] This posting is part of a current series on ontology development and tools. The series began with an update of my prior Ontology Tools listing, which now contains 185 tools. It continued with a survey of ontology development methodologies. The next part in this series will address a new architecture for tooling development. The last installment in the series is planned to cover ontology best practices. This same posting is permanently archived and

updated on the OpenStructs TechWiki as Lightweight, Domain Ontologies Development Methodology.

[2] Examples of upper-level ontologies include the Suggested Upper Merged Ontology (SUMO), the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE), PROTON, Cyc and BFO (Basic Formal Ontology). Most of the content in their upper-levels is akin to broad, abstract relations or concepts (similar to the primary classes, for example, in a Roget's Thesaurus — that is, real ontos stuff) than to "generic common knowledge." Most all of them have both a hierarchical and networked structure, though their actual subject structure relating to concrete things is generally pretty weak. For a more detailed treatment of ontology classifications, see M. K. Bergman, 2007. "An Intrepid Guide to Ontologies," *AI3:::Adaptive Information* blog, May 16, 2007.

[3] O. Corcho, M. Fernandez and A. Gomez-Perez, 2003. "Methodologies, Tools and Languages for Building Ontologies: Where is the Meeting Point?," in *Data & Knowledge Engineering* 46, 2003. See http://www.dia.fi.upm.es/~ocorcho/documents/DKE2003_CorchoEtAl.pdf.

[4] Elena Paslaru Bontas Simperl and Christoph Tempich, 2006. "Ontology Engineering: A Reality Check," in *Proceedings of the 5th International Conference on Ontologies, Databases, and Applications of Semantics ODBASE 2006*, 2006. See http://ontocom.ag-nbi.de/docs/odbase2006.pdf.

[5] OntologyDesignPatterns.org is a semantic Web portal dedicated to ontology design patterns (ODPs). The portal was started under the NeOn project, which still partly supports its development.

[6] See M.K. Bergman, 2009. "Ontology-driven Applications Using Adaptive Ontologies," *AI3:::Adaptive Information* blog, November 23, 2009.

[7] See M.K. Bergman, 2008. "The Semantics of Context," *AI3:::Adaptive Information* blog, May 6, 2008.

[8] UMBEL (*Upper Mapping and Binding Exchange Layer*) is an ontology of about 20,000 subject concepts that acts as a reference structure for inter-relating disparate datasets. It is also a general vocabulary of classes and predicates designed for the creation of domain-specific ontologies.

[9] See M.K. Bergman, 2008. "When is Content *Coherent*?," *AI3:::Adaptive Information* blog, July 25, 2008.

[10] See M.K. Bergman, 2009. "The Open World Assumption: Elephant in the Room," *AI3:::Adaptive Information* blog, December 21, 2009.

[11] MIKE2.0 (*Method for Integrated Knowledge Environments*) is an open source information development methodology championed by Bearing Point and Deloitte. Structured Dynamics has adopted the approach and has helped formulate MIKE2.0's semantic enterprise offering. For a general intro to the approach, see further M.K. Bergman, 2010. "MIKE2.0: Open Source Information Development in the Enterprise," *AI3:::Adaptive Information* blog, February 23, 2010.

[12] This is our working definition for description logics:
"Description logics and their semantics traditionally split *concepts* and their relationships from the different treatment of *instances* and their attributes and roles, expressed as fact assertions. The concept split is known as the TBox (for *terminological* knowledge, the basis for *T* in *TBox*) and represents the schema or taxonomy of the domain at hand. The TBox is the structural and intensional component of conceptual relationships. The second split of instances is known as the ABox (for *assertions*, the basis for *A* in *ABox*) and describes the attributes of instances (and individuals), the roles between instances, and other assertions about instances regarding their class membership with the TBox concepts."

[13] See the four-part description logics series from M. K. Bergman, 2009. "Making Linked Data Reasonable using Description Logics, Part 1," *AI3:::Adaptive Information* blog, Feb. 11, 2009; "Making Linked Data Reasonable using Description Logics, Part 2," *AI3:::Adaptive Information* blog, Feb. 15, 2009; "Making Linked Data Reasonable using Description Logics, Part 3," *AI3:::Adaptive Information* blog, Feb. 18, 2009; and "Making Linked Data Reasonable using Description Logics, Part 4," *AI3:::Adaptive Information* blog, Feb. 23, 2009.

[14] See Part 2 in [13].

[15] The ***TBox*** portion, or classes (concepts), is the basis of the ontologies. The ontologies establish the structure used for governing the conceptual relationships for that domain and in reference to external (Web) ontologies. The ***ABox*** portion, or instances (named entities), represents the specific, individual things that are the members of those classes. Named entities are the notable objects, persons, places, events, organizations and things of the world. Each named entity is related to one or more classes (concepts) to which it is a member. Named entities do not set the structure of the domain, but populate that structure. The ABox and TBox play different roles in the use and organization of the information and structure.

[16] The original version, now slightly modified, was first published in M. K. Bergman, 2009. "Ontology-driven Applications Using Adaptive Ontologies," *AI3:::Adaptive Information* blog, Nov. 23, 2009.

[17] As some examples, see for instance: SKOS: Mark van Assem, Veronique Malais, Alistair Miles and Guus Schreiber, 2006. "A Method to Convert Thesauri to SKOS," in *The Semantic Web: Research and Applications (2006)*, pp. 95-109. See http://www.cs.vu.nl/~mark/papers/Assem06b.pdf for paper, also http://thesauri.cs.vu.nl/eswc06/ and http://thesauri.cs.vu.nl/; taxonomies: Fausto Giunchiglia, Maurizio Marchese and Ilya Zaihrayeu, 2006. "Encoding Classifications into Lightweight Ontologies," presented at *Proceedings of the 3rd European Semantic Web Conference (ESWC 2006*), Budva. See http://www.science.unitn.it/~marchese/pdf/encoding%20classifications%20into%20lightweight%20ontologies_JoDS8.pdf; metadata: Mikael Nilsson, 2007. See http://mikaelnilsson.blogspot.com/2007/11/semanticizing-metadata-specifications.html; relational schema: see the W3C workgroup on RDB2RDF; and, of course, there are many others.

[18] Natalya F. Noy and Deborah L. McGuinness, 2001. "Ontology Development 101: A Guide to Creating Your First Ontology," Stanford University *Knowledge Systems Laboratory Technical Report KSL-01-05*, March 2001. See http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html.

[19] The various criteria that are considered in nominating an existing ontology to "core" status is that it should be general; highly used; universal; broad committee or community support; well done and documented; and easily understood.

[20] Example and comprehensive ontology editing toolkits or IDEs (integrated development environments) include NeOn toolkit, Protégé, and TopBraid Composer. A complement to these larger toolkits is the OWL API, which when used can also provide a canonical management framework for specific ontology tools and tasks. This topic is covered more in the next installment regarding the tools landscape.

[21] Good ontology design, especially for larger projects, does require a degree of modularity. An architecture of multiple ontologies often work together to isolate different work tasks so as to aid better ontology management. Ontology architecture and modularization is a separate topic in its own right.

[22] Originally published as M.K. Bergman, 2010. "An Executive Intro to Ontologies," *AI3:::Adaptive Information* blog, August 9, 2010. This popular document has now been permanently archived on the the OpenStructs TechWiki as Intro to Ontologies.

[23] Another reason for the clear distinction between ABox and TBox is their use to aid one another in disambiguation. Structured Dynamics' scones approach (***s***ubject ***c***oncepts ***o***r ***n***amed ***e***ntitie***s***) is designed expressly for this purpose. It is also possible to integrate these approaches with third-party tools (*e.g.*, Calais, Expert System (Cogito), etc.) to improve unstructured content characterization. Via this approach we now can assess concept matches in addition to entity matches. This means we can triangulate between the two assessments to aid disambiguation. Because of logical segmentation, we have increased the informational power of our concept graph.