# Ontology

## (Knowledge Representation in Information Science)

Sullivan Bowen

First Edition, 2012

ISBN 978-81-323-3091-2



*Published by:*
**Research World**
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

# Table of Contents

Chapter 1

# Ontology

In computer science and information science, an **ontology** is a formal representation of knowledge as a set of concepts within a domain, and the relationships between those concepts. It is used to reason about the entities within that domain, and may be used to describe the domain. Its meaning is vastly different from the word Ontology in philosophy.

In theory, an ontology is a "formal, explicit specification of a shared conceptualisation". An ontology provides a shared vocabulary, which can be used to model a domain — that is, the type of objects and/or concepts that exist, and their properties and relations.

Ontologies are the structural frameworks for organizing information and are used in artificial intelligence, the Semantic Web, systems engineering, software engineering, biomedical informatics, library science, enterprise bookmarking, and information architecture as a form of knowledge representation about the world or some part of it. The creation of domain ontologies is also fundamental to the definition and use of an enterprise architecture framework.

## *Overview*

The term *ontology* has its origin in philosophy, and has been applied in many different ways. The word "ontology" comes from the Greek *ὄν* (on), which literally means 'existence'. The core meaning within computer science is a model for describing the world that consists of a set of types, properties, and relationship types. Exactly what is provided around these varies, but they are the essentials of an ontology. There is also generally an expectation that there be a close resemblance between the real world and the features of the model in an ontology.

What many ontologies have in common in both computer science and in philosophy is the representation of entities, ideas, and events, along with their properties and relations,

according to a system of categories. In both fields, one finds considerable work on problems of ontological relativity (e.g., Quine and Kripke in philosophy, Sowa and Guarino in computer science), and debates concerning whether a normative ontology is viable (e.g., debates over foundationalism in philosophy, debates over the Cyc project in AI). Differences between the two are largely matters of focus. Philosophers are less concerned with establishing fixed, controlled vocabularies than are researchers in computer science, while computer scientists are less involved in discussions of first principles (such as debating whether there are such things as fixed essences, or whether entities must be ontologically more primary than processes).

## *History*

Historically, ontologies arise out of the branch of philosophy known as metaphysics, which deals with the nature of reality – of what exists. This fundamental branch is concerned with analyzing various types or modes of existence, often with special attention to the relations between particulars and universals, between intrinsic and extrinsic properties, and between essence and existence. The traditional goal of ontological inquiry in particular is to divide the world "at its joints", to discover those fundamental categories, or kinds, into which the world's objects naturally fall.

During the second half of the 20th century, philosophers extensively debated the possible methods or approaches to building ontologies, without actually *building* any very elaborate ontologies themselves. By contrast, computer scientists were building some large and robust ontologies (such as WordNet and Cyc) with comparatively little debate over *how* they were built.

Since the mid-1970s, researchers in the field of artificial intelligence (AI) have recognized that capturing knowledge is the key to building large and powerful AI systems. AI researchers argued that they could create new ontologies as computational models that enable certain kinds of automated reasoning. In the 1980s, the AI community began to use the term *ontology* to refer to both a theory of a modeled world and a component of knowledge systems. Some researchers, drawing inspiration from philosophical ontologies, viewed computational ontology as a kind of applied philosophy.

In the early 1990s, the widely cited Web page and paper "Toward Principles for the Design of Ontologies Used for Knowledge Sharing" by Tom Gruber is credited with a deliberate definition of *ontology* as a technical term in computer science. Gruber "introduced the term to mean a specification of a conceptualization. That is "an ontology is a description (like a formal specification of a program) of the concepts and relationships that can formally exist for an agent or a community of agents. This definition is consistent with the usage of ontology as set of concept definitions, but more general. And it is a different sense of the word than its use in philosophy".

According to Gruber (1993) "ontologies are often equated with taxonomic hierarchies of classes, class definitions, and the subsumption relation, but ontologies need not be limited to these forms. Ontologies are also not limited to conservative definitions – that is,

definitions in the traditional logic sense that only introduce terminology and do not add any knowledge about the world. To specify a conceptualization, one needs to state axioms that do constrain the possible interpretations for the defined terms.

In the early years of the 21st century, the interdisciplinary project of cognitive science has been bringing the two circles of scholars closer together. For example, there is talk of a "computational turn in philosophy" that includes philosophers analyzing the formal ontologies of computer science (sometimes even working directly with the software), while researchers in computer science have been making more references to those philosophers who work on ontology (sometimes with direct consequences for their methods). Still, many scholars in both fields are uninvolved in this trend of cognitive science, and continue to work independently of one another, pursuing separately their different concerns.

## *Ontology components*

Contemporary ontologies share many structural similarities, regardless of the language in which they are expressed. As mentioned above, most ontologies describe individuals (instances), classes (concepts), attributes, and relations. In this each of these components is discussed in turn.

Common components of ontologies include:

- Individuals: instances or objects (the basic or "ground level" objects)
- Classes: sets, collections, concepts, classes in programming, types of objects, or kinds of things
- Attributes: aspects, properties, features, characteristics, or parameters that objects (and classes) can have
- Relations: ways in which classes and individuals can be related to one another
- Function terms: complex structures formed from certain relations that can be used in place of an individual term in a statement
- Restrictions: formally stated descriptions of what must be true in order for some assertion to be accepted as input
- Rules: statements in the form of an if-then (antecedent-consequent) sentence that describe the logical inferences that can be drawn from an assertion in a particular form
- Axioms: assertions (including rules) in a logical form that together comprise the overall theory that the ontology describes in its domain of application. This definition differs from that of "axioms" in generative grammar and formal logic. In those disciplines, axioms include only statements asserted as *a priori* knowledge. As used here, "axioms" also include the theory derived from axiomatic statements
- Events: the changing of attributes or relations

Ontologies are commonly encoded using ontology languages.

## *Domain ontologies and upper ontologies*

A domain ontology (or domain-specific ontology) models a specific domain, or part of the world. It represents the particular meanings of terms as they apply to that domain. For example the word *card* has many different meanings. An ontology about the domain of poker would model the "playing card" meaning of the word, while an ontology about the domain of computer hardware would model the "punched card" and "video card" meanings.

An upper ontology (or foundation ontology) is a model of the common objects that are generally applicable across a wide range of domain ontologies. It employs a core glossary that contains, the terms, and associated object descriptions, as they are used in various, relevant domain sets. There are several standardized upper ontologies available for use, including Dublin Core, GFO, OpenCyc/ResearchCyc, SUMO, and DOLCE. WordNet, while considered an upper ontology by some, is not strictly an ontology. However, it has been employed as a linguistic tool for learning domain ontologies.

The Gellish ontology is an example of a combination of an upper and a domain ontology.

Since domain ontologies represent concepts in very specific and often eclectic ways, they are often incompatible. As systems that rely on domain ontologies expand, they often need to merge domain ontologies into a more general representation. This presents a challenge to the ontology designer. Different ontologies in the same domain can also arise due to different perceptions of the domain based on cultural background, education, ideology, or because a different representation language was chosen.

At present, merging ontologies that are not developed from a common foundation ontology is a largely manual process and therefore time-consuming and expensive. Domain ontologies that use the same foundation ontology to provide a set of basic elements with which to specify the meanings of the domain ontology elements can be merged automatically. There are studies on generalized techniques for merging ontologies, but this area of research is still largely theoretical.

## *Ontology engineering*

Ontology engineering (or ontology building) is a subfield of knowledge engineering that studies the methods and methodologies for building ontologies. It studies the ontology development process, the ontology life cycle, the methods and methodologies for building ontologies, and the tool suites and languages that support them.

Ontology engineering aims to make explicit the knowledge contained within software applications, and within enterprises and business procedures for a particular domain. Ontology engineering offers a direction towards solving the interoperability problems brought about by semantic obstacles, such as the obstacles related to the definitions of business terms and software classes. Ontology engineering is a set of tasks related to the development of ontologies for a particular domain.

## *Ontology languages*

An ontology language is a formal language used to encode the ontology. There are a number of such languages for ontologies, both proprietary and standards-based:

- Common Algebraic Specification Language is a general logic-based specification language developed within the IFIP working group 1.3 "Foundations of System Specifications" and functions as a de facto standard in the area of software specifications. It is now being applied to ontology specifications in order to provide modularity and structuring mechanisms.
- Common logic is ISO standard 24707, a specification for a family of ontology languages that can be accurately translated into each other.
- The Cyc project has its own ontology language called CycL, based on first-order predicate calculus with some higher-order extensions.
- DOGMA (Developing Ontology-Grounded Methods and Applications) adopts the fact-oriented modeling approach to provide a higher level of semantic stability.
- The Gellish language includes rules for its own extension and thus integrates an ontology with an ontology language.
- IDEF5 is a software engineering method to develop and maintain usable, accurate, domain ontologies.
- KIF is a syntax for first-order logic that is based on S-expressions.
- Rule Interchange Format (RIF) and F-Logic combine ontologies and rules.
- OWL is a language for making ontological statements, developed as a follow-on from RDF and RDFS, as well as earlier ontology language projects including OIL, DAML and DAML+OIL. OWL is intended to be used over the World Wide Web, and all its elements (classes, properties and individuals) are defined as RDF resources, and identified by URIs.
- Semantic Application Design Language (SADL) captures a subset of the expressiveness of OWL, using an English-like language entered via an Eclipse Plug-in.
- OBO, a language used for biological and biomedical ontologies.
- (E)MOF and UML are standards of the OMG

## *Examples of published ontologies*

- Basic Formal Ontology, a formal upper ontology designed to support scientific research
- BioPAX, an ontology for the exchange and interoperability of biological pathway (cellular processes) data
- BMO, an e-Business Model Ontology based on a review of enterprise ontologies and business model literature
- CCO (Cell Cycle Ontology), an application ontology that represents the cell cycle
- CContology (Customer Complaint Ontology), an e-business ontology to support online customer complaint management
- CIDOC Conceptual Reference Model, an ontology for cultural heritage

- COSMO, a Foundation Ontology (current version in OWL) that is designed to contain representations of all of the primitive concepts needed to logically specify the meanings of any domain entity. It is intended to serve as a basic ontology that can be used to translate among the representations in other ontologies or databases. It started as a merger of the basic elements of the OpenCyc and SUMO ontologies, and has been supplemented with other ontology elements (types, relations) so as to include representations of all of the words in the Longman dictionary defining vocabulary.
- Cyc, a large Foundation Ontology for formal representation of the universe of discourse.
- Disease Ontology, designed to facilitate the mapping of diseases and associated conditions to particular medical codes
- DOLCE, a Descriptive Ontology for Linguistic and Cognitive Engineering
- Dublin Core, a simple ontology for documents and publishing
- Foundational, Core and Linguistic Ontologies
- Foundational Model of Anatomy, an ontology for human anatomy
- Friend of a Friend, an ontology for describing persons, their activities and their relations to other people and objects
- Gene Ontology for genomics
- Gellish English dictionary, an ontology that includes a dictionary and taxonomy that includes an upper ontology and a lower ontology that focusses on industrial and business applications in engineering, technology and procurement.
- Geopolitical ontology, an ontology describing geopolitical information created by Food and Agriculture Organization(FAO). The geopolitical ontology includes names in multiple languages (English, French, Spanish, Arabic, Chinese, Russian and Italian); maps standard coding systems (UN, ISO, FAOSTAT, AGROVOC, etc); provides relations among territories (land borders, group membership, etc); and tracks historical changes.
- GOLD, General Ontology for Linguistic Description
- GUM (Generalized Upper Model), a linguistically-motivated ontology for mediating between clients systems and natural language technology
- IDEAS Group, a formal ontology for enterprise architecture being developed by the Australian, Canadian, UK and U.S. Defence Depts.
- Linkbase, a formal representation of the biomedical domain, founded upon Basic Formal Ontology.
- LPL, Lawson Pattern Language
- NIFSTD Ontologies from the Neuroscience Information Framework: a modular set of ontologies for the neuroscience domain.
- OBO Foundry, a suite of interoperable reference ontologies in biomedicine
- Ontology for Biomedical Investigations, an open access, integrated ontology for the description of biological and clinical investigations
- OMNIBUS Ontology, an ontology of learning, instruction, and instructional design
- Plant Ontology for plant structures and growth/development stages, etc.
- POPE, Purdue Ontology for Pharmaceutical Engineering

- PRO, the Protein Ontology of the Protein Information Resource, Georgetown University.
- Program abstraction taxonomy program abstraction taxonomy
- Protein Ontology for proteomics
- Suggested Upper Merged Ontology, a formal upper ontology
- Systems Biology Ontology (SBO), for computational models in biology
- SWEET, Semantic Web for Earth and Environmental Terminology
- ThoughtTreasure ontology
- TIME-ITEM, Topics for Indexing Medical Education
- UMBEL, a lightweight reference structure of 20,000 subject concept classes and their relationships derived from OpenCyc
- WordNet, a lexical reference system
- YAMATO, Yet Another More Advanced Top-level Ontology

The W3C Linking Open Data Community Project coordinates attempts to converge different ontologies into worldwide Data Web.

## *Ontology libraries*

The development of ontologies for the Web has led to the emergence of services providing lists or directories of ontologies with search facility. Such directories have been called ontology libraries.

The following are static libraries of human-selected ontologies.

- DAML Ontology Library maintains a legacy of ontologies in DAML.
- Protege Ontology Library contains a set of OWL, Frame-based and other format ontologies.
- SchemaWeb is a directory of RDF schemata expressed in RDFS, OWL and DAML+OIL.

The following are both directories and search engines. They include crawlers searching the Web for well-formed ontologies.

- OBO Foundry / Bioportal is a suite of interoperable reference ontologies in biology and biomedicine.
- OntoSelect Ontology Library offers similar services for RDF/S, DAML and OWL ontologies.
- Ontaria is a "searchable and browsable directory of semantic web data", with a focus on RDF vocabularies with OWL ontologies. (NB Project "on hold" since 2004).
- Swoogle is a directory and search engine for all RDF resources available on the Web, including ontologies.

## *Examples of applications using ontology engines*

- SAPPHIRE (Health care) or *Situational Awareness and Preparedness for Public Health Incidences and Reasoning Engines* is a semantics-based health information system capable of tracking and evaluating situations and occurrences that may affect public health.

Chapter 2

# Controlled Vocabulary

**Controlled vocabularies** provide a way to organize knowledge for subsequent retrieval. They are used in subject indexing schemes, subject headings, thesauri and taxonomies. Controlled vocabulary schemes mandate the use of predefined, authorised terms that have been preselected by the designer of the vocabulary, in contrast to natural language vocabularies, where there is no restriction on the vocabulary.

## *In library and information science*

In library and information science controlled vocabulary is a carefully selected list of words and phrases, which are used to tag units of information (document or work) so that they may be more easily retrieved by a search. Controlled vocabularies solve the problems of homographs, synonyms and polysemes by a bijection between concepts and authorized terms. In short, controlled vocabularies reduce ambiguity inherent in normal human languages where the same concept can be given different names and ensure consistency.

For example, in the Library of Congress Subject Headings (a subject heading system that uses a controlled vocabulary), authorized terms -- subject headings in this case -- have to be chosen to handle choices between variant spellings of the same concept (American versus British), choice among scientific and popular terms (Cockroaches versus Periplaneta americana), and choices between synonyms (automobile versus cars), among other difficult issues.

Choices of authorised terms are based on the principles of user warrant (what terms users are likely to use), literary warrant (what terms are generally used in the literature and documents), structural warrant (terms chosen by considering the structure, scope of the controlled vocabulary).

Controlled vocabularies also typically handle the problem of homographs, with qualifiers. For example, the term "pool" has to be qualified to refer to either swimming pool, or the game pool to ensure that each authorised term or heading refers to only one concept.

There are two main kinds of controlled vocabulary tools used in libraries: subject headings and thesauri. While the differences between the two are diminishing, there are still some minor differences.

Historically subject headings were designed to describe books in library catalogs by catalogers while thesauri were used by indexers to apply index terms to documents and articles. Subject headings tend to be broader in scope describing whole books, while thesauri tend to be more specialised covering very specific disciplines. Also because of the card catalog system, subject headings tend to have terms that are in indirect order (though with the rise of automated systems this is being removed), while thesaurus terms are always in direct order. Subject headings also tend to use more pre-co-ordination of terms such that the designer of the controlled vocabulary will combine various concepts together to form one authorised subject heading. (e.g., children and terrorism) while thesauri tend to use singular direct terms. Lastly thesauri list not only equivalent terms but also narrower, broader terms and related terms among various authorised and non-authorised terms, while historically most subject headings did not.

For example Library of Congress Subject Heading itself did not have much syndetic structure until 1943, and it was not until 1985 when it began to adopt the thesauri type term "Broader term" and "Narrow term".

The terms are chosen and organized by trained professionals (including librarians and information scientists) who possess expertise in the subject area. Controlled vocabulary terms can accurately describe what a given document is actually about, even if the terms themselves do not occur within the document's text. Well known subject heading systems include the Library of Congress system, MeSH, and Sears. Well known thesauri include the Art and Architecture Thesaurus and the ERIC Thesaurus.

Choosing authorized terms to be used is a tricky business, besides the areas already considered above, the designer has to consider the specificity of the term chosen, whether to use direct entry, inter consistency and stability of the language. Lastly the amount of pre-co-ordinate (in which case the degree of enumeration versus synthesis becomes an issue) and post co-ordinate in the system is another important issue.

Controlled vocabulary elements (terms/phrases) employed as tags, to aid in the content identification process of documents, or other information system entities (e.g. DBMS, Web Services) qualifies as metadata.

## *Indexing languages*

There are three main types of indexing languages.

- Controlled indexing language - Only approved terms can be used by the indexer to describe the document

- Natural language indexing language - Any term from the document in question can be used to describe the document.

- Free indexing language - Any term (not only from the document) can be used to describe the document.

When indexing a document, the indexer also has to choose the level of indexing exhaustivity, the level of detail in which the document is described. For example using low indexing exhaustivity, minor aspects of the work will not be described with index terms. In general the higher the indexing exhaustivity, the more terms indexed for each document.

In recent years **free text search** as a means of access to documents has become popular. This involves using natural language indexing with an indexing exhaustively set to maximum (every word in the text is *indexed*). Many studies have been done to compare the efficiency and effectiveness of free text searches against documents that have been indexed by experts using a few well chosen controlled vocabulary descriptors.

Controlled vocabularies are often claimed to improve the accuracy of free text searching, such as to reduce irrelevant items in the retrieval list. These irrelevant items (false positives) are often caused by the inherent ambiguity of natural language. Take the English word *football* for example. *Football* is the name given to a number of different team sports. Worldwide the most popular of these team sports is Association football, which also happens to be called *soccer* in several countries. The English language word football is also applied to Rugby football (Rugby union and rugby league), American football, Australian rules football, Gaelic football, and Canadian football. A search for *football* therefore will retrieve documents that are about several completely different sports. Controlled vocabulary solves this problem by tagging the documents in such a way that the ambiguities are eliminated.

Compared to free text searching, the use of a controlled vocabulary can dramatically increase the performance of an information retrieval system, if performance is measured by precision (the percentage of documents in the retrieval list that are actually relevant to the search topic).

In some cases controlled vocabulary can enhance recall as well, because unlike natural language schemes, once the correct authorised term is searched, you don't need to worry about searching for other terms that might be synonyms of that term.

However, a controlled vocabulary search may also lead to unsatisfactory recall, in that it will fail to retrieve some documents that are actually relevant to the search question.

This is particularly problematic when the search question involves terms that are sufficiently tangential to the subject area such that the indexer might have decided to tag it using a different term (but the searcher might consider the same). Essentially, this can

be avoided only by an experienced user of controlled vocabulary whose understanding of the vocabulary coincides with the way it is used by the indexer.

On the other hand free text searches have high exhaustivity (you search on every word) so it has potential for high recall (assuming you solve the problems of synonyms by entering every combination) but will have much lower precision.

Controlled vocabularies are also quickly out-dated and in fast developing fields of knowledge, the authorised terms available might not be available if they are not updated regularly. Even in the best case scenario, controlled language is often not as specific as using the words of the text itself. Indexers trying to choose the appropriate index terms might mis-interpret the author, while a free text search is in no danger of doing so, because it uses the author's own words.

The use of controlled vocabularies can be costly compared to free text searches because human experts or expensive automated systems are necessary to index each entry. Furthermore, the user has to be familiar with the controlled vocabulary scheme to make best use of the system. But as already mentioned, the control of synonyms, homographs can help increase precision.

Numerous methodologies have been developed to assist in the creation of controlled vocabularies, including faceted classification, which enables a given data record or document to be described in multiple ways.

## *Applications*

Controlled vocabularies, such as the Library of Congress Subject Headings, are an essential component of bibliography, the study and classification of books. They were initially developed in library and information science. In the 1950s, government agencies began to develop controlled vocabularies for the burgeoning journal literature in specialized fields; an example is the Medical Subject Headings (MeSH) developed by the U.S. National Library of Medicine. Subsequently, for-profit firms (called Abstracting and indexing services) emerged to index the fast-growing literature in every field of knowledge. In the 1960s, an online bibliographic database industry developed based on dialup X.25 networking. These services were seldom made available to the public because they were difficult to use; specialist librarians called search intermediaries handled the searching job. In the 1980s, the first full text databases appeared; these databases contain the full text of the index articles as well as the bibliographic information. Online bibliographic databases have migrated to the Internet and are now publicly available; however, most are proprietary and can be expensive to use. Students enrolled in colleges and universities may be able to access some of these services without charge; some of these services may be accessible without charge at a public library.

In large organizations, controlled vocabularies may be introduced to improve technical communication. The use of controlled vocabulary ensures that everyone is using the same word to mean the same thing. This consistency of terms is one of the most important

concepts in technical writing and knowledge management, where effort is expended to use the same word throughout a document or organization instead of slightly different ones to refer to the same thing.

Web searching could be dramatically improved by the development of a controlled vocabulary for describing Web pages; the use of such a vocabulary could culminate in a Semantic Web, in which the content of Web pages is described using a machine-readable metadata scheme. One of the first proposals for such a scheme is the Dublin Core Initiative. An example of a controlled vocabulary which is usable for indexing web pages is PSH.

It is unlikely that a single metadata scheme will ever succeed in describing the content of the entire Web. To create a Semantic Web, it may be necessary to draw from two or more metadata systems to describe a Web page's contents. The eXchangeable Faceted Metadata Language (XFML) is designed to enable controlled vocabulary creators to publish and share metadata systems. XFML is designed on faceted classification principles.

**Chapter 3**

# Cyc

**Cyc** is an artificial intelligence project that attempts to assemble a comprehensive ontology and knowledge base of everyday common sense knowledge, with the goal of enabling AI applications to perform human-like reasoning. The project was started in 1984 by Douglas Lenat at MCC and is developed by company Cycorp. Parts of the project are released as **OpenCyc**, which provides an API, RDF endpoint, and data dump under an open source license.

## *Overview*

The project was started in 1984 as part of Microelectronics and Computer Technology Corporation. The objective was to codify, in machine-usable form, millions of pieces of knowledge that comprise human common sense. CycL presented a proprietary knowledge representation schema that utilized first-order relationships. In 1986, Doug Lenat estimated the effort to complete Cyc would be 250,000 rules and 350 man-years of effort. The Cyc Project was spun off into Cycorp, Inc. in Austin, Texas in 1994.

The name "Cyc" (from "encyclopedia", pronounced like *syke*) is a registered trademark owned by Cycorp. The original knowledge base is proprietary, but a smaller version of the knowledge base, intended to establish a common vocabulary for automatic reasoning, was released as OpenCyc under an open source (Apache) license. More recently, Cyc has been made available to AI researchers under a research-purposes license as ResearchCyc.

Typical pieces of knowledge represented in the database are "Every tree is a plant" and "Plants die eventually". When asked whether trees die, the inference engine can draw the obvious conclusion and answer the question correctly. The Knowledge Base (KB) contains over one million human-defined assertions, rules or common sense ideas. These are formulated in the language CycL, which is based on predicate calculus and has a syntax similar to that of the Lisp programming language.

Much of the current work on the Cyc project continues to be knowledge engineering, representing facts about the world by hand, and implementing efficient inference mechanisms on that knowledge. Increasingly, however, work at Cycorp involves giving the Cyc system the ability to communicate with end users in natural language, and to assist with the knowledge formation process via machine learning.

Like many companies, Cycorp has ambitions to use the Cyc natural language understanding tools to parse the entire internet to extract structured data.

## *Knowledge base*

The concept names in Cyc are known as *constants*. Constants start with an optional "#$" and are case-sensitive. There are constants for:

- Individual items known as *individuals*, such as #$BillClinton or #$France.
- *Collections*, such as #$Tree-ThePlant (containing all trees) or #$EquivalenceRelation (containing all equivalence relations). A member of a collection is called an *instance* of that collection.
- *Truth Functions* which can be applied to one or more other concepts and return either true or false. For example #$siblings is the sibling relationship, true if the two arguments are siblings. By convention, truth function constants start with a lower-case letter. Truth functions may be broken down into logical connectives (such as #$and, #$or, #$not, #$implies), quantifiers (#$forAll, #$thereExists, etc.) and predicates.
- *Functions*, which produce new terms from given ones. For example, #$FruitFn, when provided with an argument describing a type (or collection) of plants, will return the collection of its fruits. By convention, function constants start with an upper-case letter and end with the string "Fn".

The most important predicates are #$isa and #$genls. The first one describes that one item is an instance of some collection, the second one that one collection is a subcollection of another one. Facts about concepts are asserted using certain CycL *sentences*. Predicates are written before their arguments, in parentheses:

```
(#$isa #$BillClinton #$UnitedStatesPresident)
```

"Bill Clinton belongs to the collection of U.S. presidents" and

```
(#$genls #$Tree-ThePlant #$Plant)
```

"All trees are plants".

```
(#$capitalCity #$France #$Paris)
```

"Paris is the capital of France."

Sentences can also contain variables, strings starting with "?". These sentences are called "rules". One important rule asserted about the #$isa predicate reads

```
(#$implies
   (#$and
     (#$isa ?OBJ ?SUBSET)
     (#$genls ?SUBSET ?SUPERSET))
   (#$isa ?OBJ ?SUPERSET))
```

with the interpretation "if OBJ is an instance of the collection SUBSET and SUBSET is a subcollection of SUPERSET, then OBJ is an instance of the collection SUPERSET". Another typical example is

```
 (#$relationAllExists #$biologicalMother #$ChordataPhylum
#$FemaleAnimal)
```

which means that for every instance of the collection #$ChordataPhylum (i.e. for every chordate), there exists a female animal (instance of #$FemaleAnimal) which is its mother (described by the predicate #$biologicalMother).

The knowledge base is divided into *microtheories* (Mt), collections of concepts and facts typically pertaining to one particular realm of knowledge. Unlike the knowledge base as a whole, each microtheory is required to be free from contradictions. Each microtheory has a name which is a regular constant; microtheory constants contain the string "Mt" by convention. An example is #$MathMt, the microtheory containing mathematical knowledge. The microtheories can inherit from each other and are organized in a hierarchy: one specialization of #$MathMt is #$GeometryGMt, the microtheory about geometry.

## *Inference engine*

An inference engine is a computer program that tries to derive answers from a knowledge base. The Cyc inference engine performs general logical deduction (including modus ponens, modus tollens, universal quantification and existential quantification).

## *Releases*

### OpenCyc

The latest version of OpenCyc, 2.0, was released in July 2009. OpenCyc 1.0 includes the entire Cyc ontology containing hundreds of thousands of terms, along with millions of assertions relating the terms to each other; however, these are mainly taxonomic assertions, not the complex rules available in Cyc. The knowledge base contains 47,000 concepts and 306,000 facts and can be browsed on the OpenCyc website.

The first version of OpenCyc was released in spring 2002 and contained only 6,000 concepts and 60,000 facts. The knowledge base is released under the Apache License.

Cycorp has stated its intention to release OpenCyc under parallel, unrestricted licences to meet the needs of its users. The CycL and SubL interpreter (the program that allows you to browse and edit the database as well as to draw inferences) is released free of charge, but only as a binary, without source code. It is available for Linux and Microsoft Windows. The open source Texai project has released the RDF-compatible content extracted from OpenCyc.

### ResearchCyc

In July 2006, Cycorp released the binaries of ResearchCyc 1.0, a version of Cyc aimed at the research community, at no charge. (ResearchCyc was in beta stage of development during all of 2004; a beta version was released in February 2005.) In addition to the taxonomic information contained in OpenCyc, ResearchCyc includes significantly more semantic knowledge (i.e., additional facts) about the concepts in its knowledge base, and includes a large lexicon, English parsing and generation tools, and Java based interfaces for knowledge editing and querying.

## *Applications*

### Terrorism Knowledge Base

The comprehensive Terrorism Knowledge Base is an application of Cyc in development that will try to ultimately contain all relevant knowledge about "terrorist" groups, their members, leaders, ideology, founders, sponsors, affiliations, facilities, locations, finances, capabilities, intentions, behaviors, tactics, and full descriptions of specific terrorist events. The knowledge is stored as statements in mathematical logic, suitable for computer understanding and reasoning.

### Cleveland Clinic Foundation

The Cleveland Clinic has used Cyc to develop a natural language query interface of biomedical information. The query is parsed into a set of **CycL** (higher-order logic) fragments with open variables, then after applying various constraints (medical domain knowledge, common sense, discourse pragmatics, syntax), there is a way to fit those fragments together, one semantically meaningful formal query.

## *Criticisms of the Cyc Project*

The Cyc project has been described as "one of the most controversial endeavors of the artificial intelligence history", so it has inevitably garnered its share of criticism. Criticisms include:

- The complexity of the system—arguably necessitated by its encyclopedic ambitions—and the consequent difficulty in adding to the system by hand
- Scalability problems, from widespread reification, especially as constants

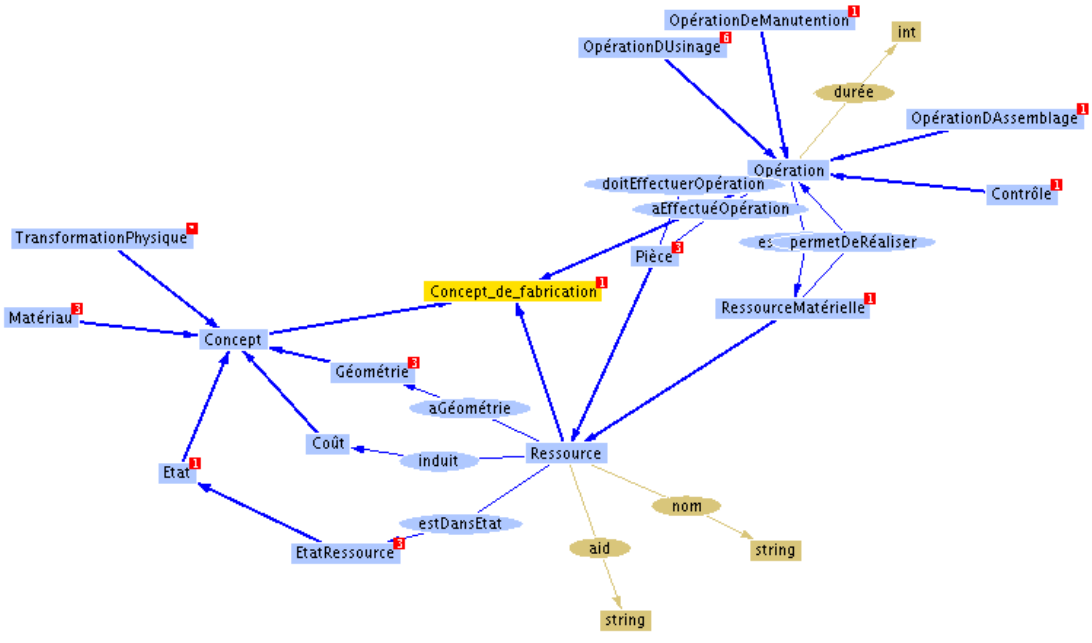
- Unsatisfactory treatment of the concept of substance and the related distinction between intrinsic and extrinsic properties
- The lack of any meaningful benchmark or comparison for the efficiency of Cyc's inference engine
- The current incompleteness of the system in both breadth and depth and the related difficulty in measuring its completeness
- Limited documentation
- The lack of up-to-date on-line training material makes it difficult for new people to learn the systems
- A large number of gaps in not only the ontology of ordinary objects, but an almost complete lack of relevant assertions describing such objects

Chapter 4

# Ontological Translator

An **ontological translator** translates a word, phrase, or grammatical construction from any language (including but not exclusive to computer language) into that of another langauge. Essentially, the procedure of a computer ontological translator is an automation of the mappings already determined between two languages. The translator is responsible for turning the extracted entities of one expression from a language into an attractive and easy to read form of another language.

## *Ontology*




Example of an ontology visualized: the Mason-ontology.

Ontology, in analytic philosophy, concerns the determining of whether some *categories of being* are fundamental and asks in what sense the items in those categories can be said to "be". It is the inquiry into being *in so much as* it is being, or into beings insofar as they exist—and not insofar as, for instance, particular facts obtained about them or particular properties related to them.

In computer science and information science, an **ontology** is a formal representation of knowledge as a set of concepts within a domain, and the relationships between those concepts. It is used to reason about the entities within that domain, and may be used to describe the domain.

In theory, an ontology is a "formal, explicit specification of a shared conceptualisation". An ontology provides a shared vocabulary, which can be used to model a domain — that is, the type of objects and/or concepts that exist, and their properties and relations..

Ontologies are the structural frameworks for organizing information and are used in artificial intelligence, the Semantic Web, systems engineering, software engineering, biomedical informatics, library science, enterprise bookmarking, and information architecture as a form of knowledge representation about the world or some part of it. The creation of domain ontologies is also fundamental to the definition and use of an enterprise architecture framework.

What ontology has in common in both computer science and in philosophy is the representation of entities, ideas, and events, along with their properties and relations, according to a system of categories. In both fields, one finds considerable work on problems of ontological relativity (e.g., Quine and Kripke in philosophy, Sowa and Guarino in computer science), and debates concerning whether a normative ontology is viable (e.g., debates over foundationalism in philosophy, debates over the Cyc project in AI). Differences between the two are largely matters of focus. Philosophers are less concerned with establishing fixed, controlled vocabularies than are researchers in computer science, while computer scientists are less involved in discussions of first principles (such as debating whether there are such things as fixed essences, or whether entities must be ontologically more primary than processes).

## *Ontology language*

In computer science and artificial intelligence, 'ontology languages' are formal languages used to construct ontologies. They allow the encoding of knowledge about specific domains and often include reasoning rules that support the processing of that knowledge. Ontology languages are usually declarative languages, are almost always generalizations of frame languages, and are commonly based on either first-order logic or on description logic.

## *Translation*

'Translation' is the communication of the meaning of a source-language text by means of an equivalent target-language text.

Translators always risk inappropriate spill-over of source-language idiom and usage into the target-language translation. On the other hand, spill-overs have imported useful source-language calques and loanwords that have enriched the target languages. Indeed, translators have helped substantially to shape the languages into which they have translated.

Because of the laboriousness of translation, since the 1940s engineers have sought to automate translation (machine translation) or to mechanically aid the human translator (computer-assisted translation). The rise of the Internet has fostered a world-wide market for translation services and has facilitated language localization.

## *Sentence classifier*

A sentence classifier is a module responsible for examining and classifying chunks of natural language text. From the crude description received, the sentence classifier divides the description into sentences, then individually examines and automatically classifies into a set of predefined classes. A sentence can belong to one or more classes, or none of them.

## *Entity extractor*

With sentences classified and grouped by types, the entity extractor module tries to extract useful entities by

1. finding useful entities and
2. dealing with misspelled words.

## *Ontology instantiator*

The ontology instantiator

1. maintains the entities extracted,
2. provides the basic structure and organization of the involved concepts,
3. considers possible semantic relationships between objects and attributes, and
4. generates an instance that contains the entities, their attributes, and their semantic relationships.

This information is then fed into the ontological translator for production of an attractive and easy to read form in the target language.

Chapter 5

# Ontology Alignment and Ontology Chart

## Ontology Alignment

**Ontology alignment**, or **ontology matching**, is the process of determining correspondences between concepts. A set of correspondences is also called an alignment. The phrase takes on a slightly different meaning, in computer science, cognitive science or philosophy.

### *Computer Science*

For computer scientists, concepts are expressed as labels for data. Historically, the need for ontology alignment arose out of the need to integrate hetereogeneous databases, ones developed independently and thus each having their own data vocabulary. In the Semantic Web context involving many actors providing their own ontologies, ontology matching has taken a critical place for helping heterogeneous resources to interoperate. Ontology alignment tools find classes of data that are "semantically equivalent," for example, "Truck" and "Lorry." The classes are not necessarily logically identical. According to Euzenat and Shvaiko (2007), there are three major dimensions for similarity: syntactic, external, and semantic. Coincidentally, they roughly correspond to the dimensions identified by Cognitive Scientists below. A number of tools and frameworks have been developed for aligning ontologies, some with inspiration from Cognitive Science and some independently.

Ontology alignment tools have generally been developed to operate on database schemas, XML schemas, taxonomies, formal languages, entity-relationship models, dictionaries, and other label frameworks. They are usually converted to a graph representation before being matched. Since the emergence of the Semantic Web, such graphs can be represented in the Resource Description Framework line of languages by triples of the form <subject, predicate, object>, as illustrated in the Notation 3 syntax. In this context, aligning ontologies is sometimes referred to as "ontology matching".

The problem of Ontology Alignment has been tackled recently by trying to compute matching first and mapping (based on the matching) in an automatic fashion. Systems like DSSim, X-SOM or COMA++ obtained at the moment very high precision and recall. The Ontology Alignment Evaluation Initiative aims to evaluate, compare and improve the different approaches.

More recently, a technique useful to minimize the effort in mapping validation and visualization has been presented which is based on Minimal Mappings. Minimal mappings are high quality mappings such that i) all the other mappings can be computed from them in time linear in the size of the input graphs, and ii) none of them can be dropped without losing property i).

## Formal Definition

Given two ontologies $i = \langle C_i, R_i, I_i, A_i \rangle$and $j = \langle C_j, R_j, I_j, A_j \rangle$we can define different type of (inter-ontology) relationships among their terms. Such relationships will be called, all together, alignments and can be categorized among different dimensions:

- similarity vs logic: this is the difference between matchings (predicating about the similarity of ontology terms), and mappings (logical axioms, typically expressing logical equivalence or inclusion among ontology terms)
- atomic vs complex: whether the alignments we considered are one-to-one, or can involve more terms in a query-like formulation (e.g., LAV/GAV mapping)
- homogeneous vs heterogeneous: do the alignments predicate on terms of the same type (e.g., classes are related only to classes, individuals to individuals, etc.) or we allow heterogeneity in the relationship?
- type of alignment: the semantics associated to an alignment. It can be subsumption, equivalence, disjointness, part-of or any user-specify relationship.

Subsumption, atomic, homogeneous alignments are the building blocks to obtain richer alignments, and have a well defined semantics in every Description Logic. Let's now introduce more formally ontology matching and mapping.

An atomic homogeneous **matching** is an alignment that carries a similarity degree $s \in [0, 1]$, describing the similarity of two terms of the input ontologies *i* and *j*. Matching can be both *computed*, by means heuristic algorithms, or *inferred* from other matchings.

Formally we can say that, a matching is a quadruple $m = \langle id, t_i, t_j, s \rangle$, where $t_i$ and $t_j$ are homogeneous ontology terms, *s* is the similarity degree of *m*. A (subsumption, homogeneous, atomic) mapping is defined as a pair $\mu = \langle t_i, t_j \rangle$, where $t_i$ and $t_j$ are homogeneous ontology terms.

### *Cognitive Science*

For cognitive scientists interested in ontology alignment, the "concepts" are nodes in a semantic network that reside in brains as "conceptual systems." The focal question is: if everyone has unique experiences and thus different semantic networks, then how can we ever understand each other? This question has been addressed by a model called ABSURDIST (Aligning Between Systems Using Relations Derived Inside Systems for Translation). Three major dimensions have been identified for similarity as equations for "internal similarity, external similarity, and mutual inhibition."

Ontology alignment is closely related to analogy formation, where "concepts" are variables in logic expressions.

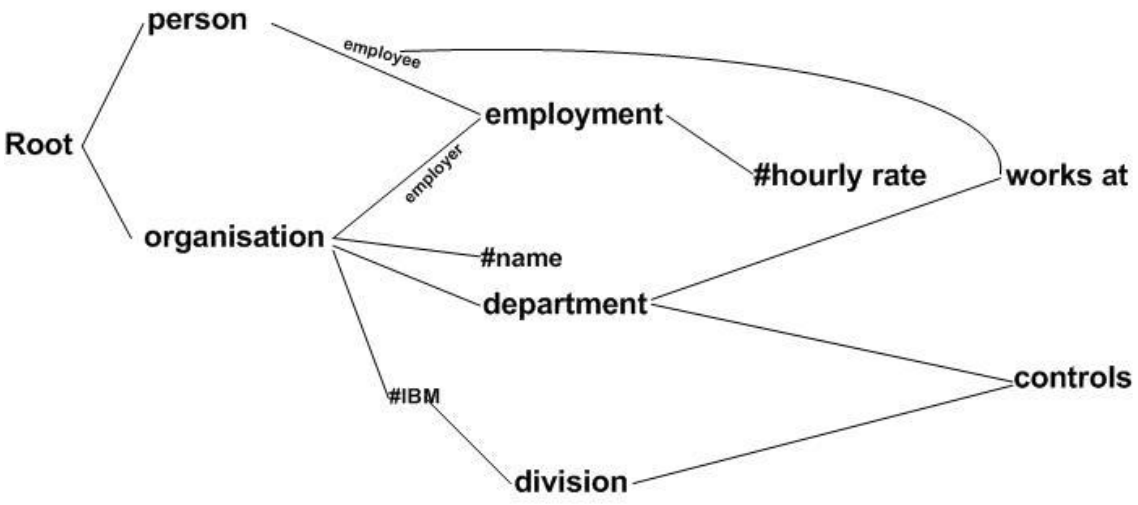
### *Philosophy*

For philosophers, much like cognitive scientists, the interest is in the nature of "understanding." The roots of discourse, however, may be traced to radical interpretation.

### *Visualization Tools*

- Optima: A Visual Ontology Alignment Tool
- CogZ: Cognitive Support and Visualization for Human-Guided Mapping Systems
- AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies

# Ontology Chart




The case study used here to illustrate the transformation process is a slight adaptation of the case used by Bonacin (2004).

An **Ontology Chart** is a type of chart used in Semiotics and software engineering to illustrate an ontology.

## *Overview*

The nodes of an Ontology chart represent universal affordances and rarely represent particulars. The exception is the Root which is a particular agent often labelled 'Society' and located on the extreme left of an Ontology chart. The Root is often dropped in practice but is implied in every Ontology chart. If any other particular is present in an Ontology chart it is recognised by the '#' sign prefix and upper case letters. In our Ontology chart the node labelled #IBM is a particular organisation.

The arcs represent ontological dependency relations directed from left to right. The right affordance is ontologically dependent on the left affordance. The left affordance is the ontological antecedent of the right affordance. A special category of affordances are determiners. They are recognised by the '#' sign prefix. The two examples above are #hourly rate and #name. All determiners have a second antecedent – the measurement standard. They are usually dropped from the Ontology chart but they are implied and obvious. In the case of hourly rate and name they are currency and language respectively. The names on the arcs are role names of the carrier, the left node, in the relationship node on the right. For example, the 'employee' is the role name of a person while in employment. No Ontology chart node has more than two ontological antecedents. Where you find an arc on the ontology chart between a role name and a node, read that as an arc between the right hand side of the role name. So the arc from employee to works at is an arc between employment and works at

Mathematically, Ontology charts are a graphical representation of semi-lattice structures; specifically they are Hasse diagrams of a single root and no cycles. Ontological dependency is a relationship known mathematically as a partial order set relation (poset). Posets are an object of study in the mathematical discipline of order theory. They belong to the class of binary relations but they have three additional properties: reflexivity, anti-symmetry and transitivity.

Ontological dependency is a poset because it is a binary relation, every thing is ontologically dependent on itself for its existence, two things that are mutually ontologically dependent must be the same thing and if a depends on b and b depends on c then a depends on c. The last of these properties - the transitive property of posets - was exploited by Helmut Hasse to give us the Hasse diagram - a diagram of incredible power, simplicity and if drawn well elegant as well. Because Ontology charts have a Root that all affordances (realisations/things) are ultimately dependent upon for their existence, they are graphical representation of semi-lattices.

**Chapter 6**

# Ontology Components

Contemporary ontologies share many structural similarities, regardless of the language in which they are expressed. Most ontologies describe individuals (instances), classes (concepts), attributes, and relations.

## *Overview*

Common components of ontologies include:

- Individuals: instances or objects (the basic or "ground level" objects)
- Classes: sets, collections, concepts, types of objects, or kinds of things.
- Attributes: aspects, properties, features, characteristics, or parameters that objects (and classes) can have
- Relations: ways in which classes and individuals can be related to one another
- Function terms: complex structures formed from certain relations that can be used in place of an individual term in a statement
- Restrictions: formally stated descriptions of what must be true in order for some assertion to be accepted as input
- Rules: statements in the form of an if-then (antecedent-consequent) sentence that describe the logical inferences that can be drawn from an assertion in a particular form
- Axioms: assertions (including rules) in a logical form that together comprise the overall theory that the ontology describes in its domain of application. This definition differs from that of "axioms" in generative grammar and formal logic. In these disciplines, axioms include only statements asserted as *a priori* knowledge. As used here, "axioms" also include the theory derived from axiomatic statements.
- Events: the changing of attributes or relations

Ontologies are commonly encoded using ontology languages.

## Individuals

Individuals (instances) are the basic, "ground level" components of an ontology. The individuals in an ontology may include concrete objects such as people, animals, tables, automobiles, molecules, and planets, as well as abstract individuals such as numbers and words (although there are differences of opinion as to whether numbers and words are classes or individuals). Strictly speaking, an ontology need not include any individuals, but one of the general purposes of an ontology is to provide a means of classifying individuals, even if those individuals are not explicitly part of the ontology.

In formal extensional ontologies, only the utterances of words and numbers are considered individuals – the numbers and names themselves are classes. In a 4D ontology, an individual is identified by its spatio-temporal extent. Examples of formal extensional ontologies are ISO 15926 and the model in development by the IDEAS Group.

## Classes

Classes – concepts that are also called *type*, *sort*, *category*, and *kind* – can be defined as an extension or an intension. According to an extensional definition, they are abstract groups, sets, or collections of objects. According to an intensional definition, they are abstract objects that are defined by values of aspects that are constraints for being member of the class. The first definition of class results in ontologies in which a class is a subclass of collection. The second definition of class results in ontologies in which collections and classes are more fundamentally different. Classes may classify individuals, other classes, or a combination of both. Some examples of classes:

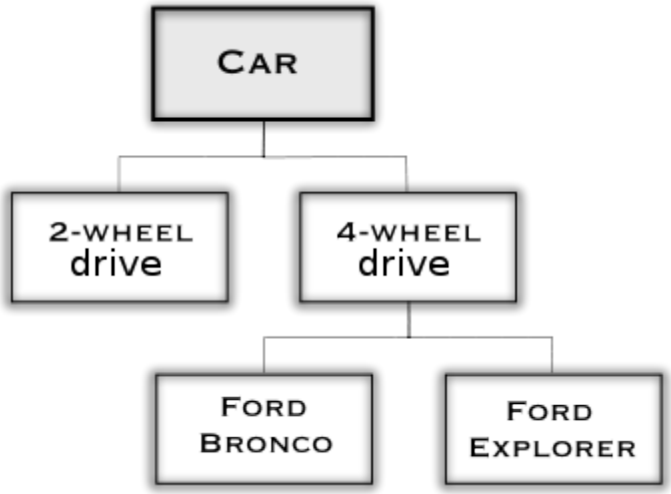
- *Person*, the class of all people, or the abstract object that can be described by the criteria for being a person.
- *Vehicle*, the class of all vehicles, or the abstract object that can be described by the criteria for being a vehicle.
- *Car*, the class of all cars, or the abstract object that can be described by the criteria for being a car.
- *Class*, representing the class of all classes, or the abstract object that can be described by the criteria for being a class.
- *Thing*, representing the class of all things, or the abstract object that can be described by the criteria for being a thing (and not nothing).

Ontologies vary on whether classes can contain other classes, whether a class can belong to itself, whether there is a universal class (that is, a class containing everything), etc. Sometimes restrictions along these lines are made in order to avoid certain well-known paradoxes.

The classes of an ontology may be extensional or intensional in nature. A class is extensional if and only if it is characterized solely by its membership. More precisely, a class C is extensional if and only if for any class C', if C' has exactly the same members

as C, then C and C' are identical. If a class does not satisfy this condition, then it is intensional. While extensional classes are more well-behaved and well-understood mathematically, as well as less problematic philosophically, they do not permit the fine grained distinctions that ontologies often need to make. For example, an ontology may want to distinguish between the class of all creatures with a kidney and the class of all creatures with a heart, even if these classes happen to have exactly the same members. In most upper ontologies, the classes are defined intensionally. Intensionally defined classes usually have necessary conditions associated with membership in each class. Some classes may also have sufficient conditions, and in those cases the combination of necessary and sufficient conditions make that class a fully *defined* class.

Importantly, a class can subsume or be subsumed by other classes; a class subsumed by another is called a *subclass* (or *subtype*) of the subsuming class (or *supertype*). For example, *Vehicle* subsumes *Car*, since (necessarily) anything that is a member of the latter class is a member of the former. The subsumption relation is used to create a hierarchy of classes, typically with a maximally general class like *Anything* at the top, and very specific classes like *2002 Ford Explorer* at the bottom. The critically important consequence of the subsumption relation is the inheritance of properties from the parent (subsuming) class to the child (subsumed) class. Thus, anything that is necessarily true of a parent class is also necessarily true of all of its subsumed child classes. In some ontologies, a class is only allowed to have one parent (*single inheritance*), but in most ontologies, classes are allowed to have any number of parents (*multiple inheritance*), and in the latter case all necessary properties of each parent are inherited by the subsumed child class. Thus a particular class of animal (*HouseCat*) may be a child of the class *Cat* and also a child of the class *Pet*.




A partition is a set of related classes and associated rules that allow objects to be classified by the appropriate subclass. The rules correspond with the aspect values that distinguish the subclasses from the superclasses. For example, to the right is the partial diagram of an ontology that has a partition of the *Car* class into the classes *2-Wheel Drive*

*Car* and *4-Wheel Drive Car*. The partition rule (or subsumption rule) determines if a particular car is classified by the *2-Wheel Drive Car* or the *4-Wheel Drive Car* class.

If the partition rule(s) guarantee that a single *Car* cannot be in both classes, then the partition is called a disjoint partition. If the partition rules ensure that every concrete object in the super-class is an instance of at least one of the partition classes, then the partition is called an exhaustive partition.

## Attributes

Objects in an ontology can be described by relating them to other things, typically aspects or parts. These related things are often called *attributes*, although they may be independent things. Each attribute can be a class or an individual. The kind of object and the kind of attribute determine the kind of relation between them. A relation between an object and an attribute express a fact that is specific to the object to which it is related. For example the Ford Explorer object has attributes such as:

- <has as name> Ford Explorer
- <has by definition as part> *door* (with as minimum and maximum cardinality: 4)
- <has by definition as part one of> *{4.0L engine, 4.6L engine}*
- <has by definition as part> *6-speed transmission*

The value of an attribute can be a complex data type; in this example, the related engine can only be one of a list of subtypes of engines, not just a single thing.

Ontologies are only true ontologies if concepts are related to other concepts (the concepts do have attributes). If that is not the case, then you would have either a taxonomy (if hyponym relationships exist between concepts) or a controlled vocabulary. These are useful, but are not considered true ontologies.

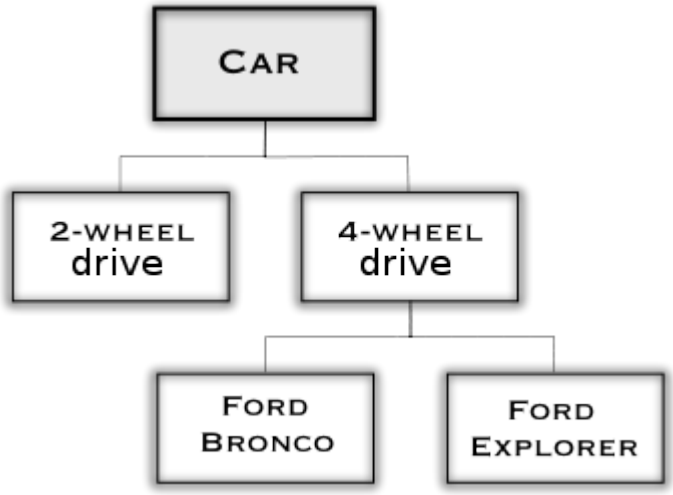
## Relationships

Relationships (also known as relations) between objects in an ontology specify how objects are related to other objects. Typically a relation is of a particular type (or class) that specifies in what sense the object is related to the other object in the ontology. For example in the ontology that contains the concept Ford Explorer and the concept Ford Bronco might be related by a relation of type <is defined as a successor of>. The full expression of that fact then becomes:

- Ford Explorer *is defined as a successor of* : Ford Bronco

This tells us that the Explorer is the model that replaced the Bronco. This example also illustrates that the relation has a direction of expression. The inverse expression expresses the same fact, but with a reverse phrase in natural language.

Much of the power of ontologies comes from the ability to describe relations. Together, the set of relations describes the semantics of the domain. The set of used relation types (classes of relations) and their subsumption hierarchy describe the expression power of the language in which the ontology is expressed.




The most important type of relation is the subsumption relation (*is-a-superclass-of*, the converse of *is-a*, *is-a-subtype-of* or *is-a-subclass-of*). This defines which objects are classified by which class. For example we have already seen that the class Ford Explorer *is-a-subclass-of* 4-Wheel Drive Car, which in turn *is-a-subclass-of* Car.

The addition of the is-a-subclass-of relationships creates a hierarchical taxonomy; a tree-like structure (or, more generally, a partially ordered set) that clearly depicts how objects relate to one another. In such a structure, each object is the 'child' of a 'parent class' (Some languages restrict the is-a-subclass-of relationship to one parent for all nodes, but many do not).

Another common type of relations is the meronymy relation, written as *part-of*, that represents how objects combine together to form composite objects. For example, if we extended our example ontology to include concepts like Steering Wheel, we would say that a "Steering Wheel is-by-definition-a-part-of-a Ford Explorer" since a steering wheel is always one of the components of a Ford Explorer. If we introduce meronymy relationships to our ontology, the hierarchy that emerges is no longer able to be held in a simple tree-like structure since now members can appear under more than one parent or branch. Instead this new structure that emerges is known as a directed acyclic graph.

As well as the standard is-a-subclass-of and is-by-definition-a-part-of-a relations, ontologies often include additional types of relations that further refine the semantics they model. Ontologies might distinguish between different categories of relation types. For example:

- relation types for relations between classes
- relation types for relations between individuals
- relation types for relations between an individual and a class
- relation types for relations between a single object and a collection
- relation types for relations between collections

Relation types are sometimes domain-specific and are then used to store specific kinds of facts or to answer particular types of questions. If the definitions of the relation types are included in an ontology, then the ontology defines its own ontology definition language. An example of an ontology that defines its own relation types and distinguishes between various categories of relation types is the Gellish ontology.

For example in the domain of automobiles, we might need a *made-in* type relationship which tells us where each car is built. So the Ford Explorer is *made-in* Louisville. The ontology may also know that Louisville is-located-in Kentucky and Kentucky is-classified-as-a state and is-a-part-of the U.S.. Software using this ontology could now answer a question like "which cars are made in the U.S.?"

Chapter 7

# Formal Concept Analysis

**Formal concept analysis** is a principled way of automatically deriving an ontology from a collection of objects and their properties. The term was introduced by Rudolf Wille in 1984, and builds on applied lattice and order theory that was developed by Birkhoff and others in the 1930s.

## *Intuitive description*

Formal concept analysis refers to both an unsupervised machine learning technique and, more broadly, a method of data analysis. The approach takes as input a matrix specifying a set of objects and the properties thereof, called attributes, and finds both all the "natural" clusters of attributes and all the "natural" clusters of objects in the input data, where

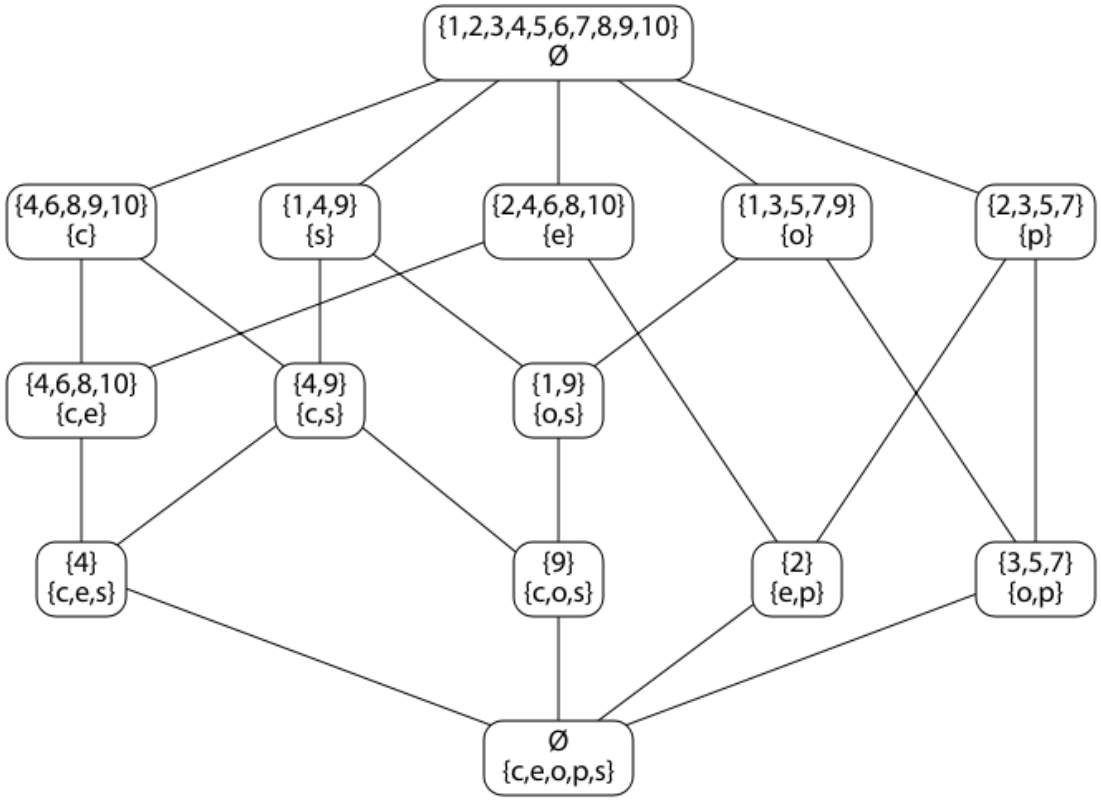
- a "natural" *object* cluster is the set of all objects that share a common subset of attributes, and
- a "natural" *property* cluster is the set of all attributes shared by one of the natural object clusters.

Natural property clusters correspond one-for-one with natural object clusters, and a **concept** is a pair containing both a natural property cluster and its corresponding natural object cluster. The family of these concepts obeys the mathematical axioms defining a lattice, and is called a **concept lattice** (in French this is called a **Treillis de Galois** because the relation between the sets of concepts and attributes is a Galois connection).

Note the strong parallel between "natural" property clusters and definitions in terms of individually necessary and jointly sufficient conditions, on one hand, and between "natural" object clusters and the extensions of such definitions, on the other. Provided the input objects and input concepts provide a complete description of the world (never true in practice, but perhaps a reasonable approximation), then the set of attributes in each concept can be interpreted as a set of singly necessary and jointly sufficient conditions for defining the set of objects in the concept. Conversely, if a set of attributes is *not* identified

as a concept in this framework, then those attributes are not singly necessary and jointly sufficient for defining *any* non-empty subset of objects in the world.

## *Example*




A concept lattice for objects consisting of the integers from 1 to 10, and attributes composite (c), square (s), even (e), odd (o) and prime (p). The lattice is drawn as a Hasse diagram.

Consider $O$ = {1,2,3,4,5,6,7,8,9,10}, and $A$ = {composite, even, odd, prime, square}. The smallest concept including the number 3 is the one with objects {3,5,7}, and attributes {odd, prime}, for 3 has both of those attributes and {3,5,7} is the set of objects having that set of attributes. The largest concept involving the attribute of being square is the one with objects {1,4,9} and attributes {square}, for 1, 4 and 9 are all the square numbers and all three of them have that set of attributes. It can readily be seen that both of these example concepts satisfy the formal definitions below

The full set of concepts for these objects and attributes is shown in the illustration. It includes a concept for each of the original attributes: the composite numbers, square numbers, even numbers, odd numbers, and prime numbers. Additionally it includes concepts for the even composite numbers, composite square numbers (that is, all square numbers except 1), even composite squares, odd squares, odd composite squares, even primes, and odd primes.

## *Contexts and concepts*

A *(formal) context* consists of a set of objects $O$, a set of attributes $A$, and an indication of which objects have which attributes. Formally it can be regarded as a bipartite graph $I \subseteq O \times A$.

| | composite | even | odd | prime | square |
|---|---|---|---|---|---|
| **1** | | | √ | | √ |
| **2** | | √ | | √ | |
| **3** | | | √ | √ | |
| **4** | √ | √ | | | √ |
| **5** | | | √ | √ | |
| **6** | √ | √ | | | |
| **7** | | | √ | √ | |
| **8** | √ | √ | | | |
| **9** | √ | | √ | | √ |
| **10** | √ | √ | | | |

A *(formal) concept* for a context is defined to be a pair $(O_i, A_i)$ such that

1. $O_i \subseteq O$
2. $A_i \subseteq A$
3. every object in $O_i$ has every attribute in $A_i$
4. for every object in $O$ that is not in $O_i$, there is an attribute in $A_i$ that the object does not have
5. for every attribute in $A$ that is not in $A_i$, there is an object in $O_i$ that does not have that attribute

$O_i$ is called the *extent* of the concept, $A_i$ the *intent*.

A context may be described as a table, with the objects corresponding to the rows of the table, the attributes corresponding to the columns of the table, and a Boolean value (in the example represented graphically as a checkmark) in cell $(x, y)$ whenever object $x$ has value $y$:

A concept, in this representation, forms a maximal subarray (not necessarily contiguous) such that all cells within the subarray are checked. For instance, the concept highlighted with a different background color in the example table is the one describing the odd prime numbers, and forms a $3 \times 2$ subarray in which all cells are checked.

## Concept lattice of a context

The concepts $(O_i, A_i)$ defined above can be partially ordered by inclusion: if $(O_i, A_i)$ and $(O_j, A_j)$ are concepts, we define a partial order $\leq$ by saying that $(O_i, A_i) \leq (O_j, A_j)$ whenever $O_i \subseteq O_j$. Equivalently, $(O_i, A_i) \leq (O_j, A_j)$ whenever $A_j \subseteq A_i$.

Every pair of concepts in this partial order has a unique greatest lower bound (meet). The greatest lower bound of $(O_i, A_i)$ and $(O_j, A_j)$ is the concept with objects $O_i \cap O_j$; it has as its attributes the union of $A_i$, $A_j$, and any additional attributes held by all objects in $O_i \cap O_j$. Symmetrically, every pair of concepts in this partial order has a unique least upper bound (join). The least upper bound of $(O_i, A_i)$ and $(O_j, A_j)$ is the concept with attributes $A_i \cap A_j$; it has as its objects the union of $O_i$, $O_j$, and any additional objects that have all attributes in $A_i \cap A_j$.

These meet and join operations satisfy the axioms defining a lattice. In fact, by considering infinite meets and joins, analogously to the binary meets and joins defined above, one sees that this is a complete lattice. Conversely, any finite lattice may be generated as the concept lattice for some context. For, let $L$ be a finite lattice, and form a context in which the objects and the attributes both correspond to elements of $L$. In this context, let object $x$ have attribute $y$ exactly when $x$ and $y$ are ordered as $x \leq y$ in the lattice. Then, the concept lattice of this context is isomorphic to $L$ itself. This construction may be interpreted as forming the Dedekind–MacNeille completion of $L$, which is known to produce an isomorphic lattice from any finite lattice.

## Concept algebra of a context

Modelling negation in a formal context is somewhat problematic because the complement $(O \backslash O_i, A \backslash A_i)$ of a concept $(O_i, A_i)$ is in general not a concept. However, since the concept lattice is complete one can consider the join $(O_i, A_i)^{\Delta}$ of all concepts $(O_j, A_j)$ that satisfy $O_j \subseteq G \backslash O_i$; or dually the meet $(O_i, A_i)^{\square}$ of all concepts satisfying $A_j \subseteq G \backslash A_i$. These two operations are known as *weak negation* and *weak opposition*, respectively.

This can be expressed in terms of the *derivative* functions. The derivative of a set $O_i \subseteq O$ of objects is the set $O_i' \subseteq A$ of all attributes that hold for all objects in $O_i$. The derivative of a set $A_i \subseteq A$ of attributes is the set $A_i' \subseteq O$ of all objects that have all attributes in $A_i$. A pair $(O_i, A_i)$ is a concept if and only if $O_i' = A_i$ and $A_i' = O_i$. Using this function, weak negation can be written as

$$(O_i, A_i)^{\Delta} = ((G \backslash A)'', (G \backslash A)'),$$

and weak opposition can be written as

$$(O_i, A_i)^{\square} = ((M \backslash B)', (M \backslash B)'').$$

The concept lattice equipped with the two additional operations Δ and □ is known as the *concept algebra* of a context. Concept algebras are a generalization of power sets.

Weak negation on a concept lattice $L$ is a *weak complementation*, i.e. an order-reversing map $\Delta: L \to L$ which satisfies the axioms $x^{\Delta\Delta} \leq x$ and $(x \wedge y) \vee (x \wedge y^{\Delta}) = x$. Weak composition is a dual weak complementation. A (bounded) lattice such as a concept algebra, which is equipped with a weak complementation and a dual weak complementation, is called a *weakly dicomplemented lattice*. Weakly dicomplemented lattices generalize distributive orthocomplemented lattices, i.e. Boolean algebras.

## Recovering the context from the Hasse diagram

The Hasse diagram of the concept lattice (also called, in formal concept analysis, a *line diagram*), encodes enough information to recover the original context from which it was formed. Each object of the context corresponds to a lattice element, the element with the minimal object set that contains that object, and with an attribute set consisting of all attributes of the object. Symmetrically, each attribute of the context corresponds to a lattice element, the one with the minimal attribute set containing that attribute, and with an object set consisting of all objects with that attribute. We may label the nodes of the Hasse diagram with the objects and attributes they correspond to; with this labeling, object $x$ has attribute $y$ if and only if there exists a monotonic path from $x$ to $y$ in the diagram.

## Efficient construction

Kuznetsov & Obiedkov (2001) survey the many algorithms that have been developed for constructing concept lattices. These algorithms vary in many details, but are in general based on the idea that each edge of the Hasse diagram of the concept lattice connects some concept $C$ to the concept formed by the join of $C$ with a single object. Thus, one can build up the concept lattice one concept at a time, by finding the neighbors in the Hasse diagram of known concepts, starting from the concept with an empty set of objects. The amount of time spent to traverse the entire concept lattice in this way is polynomial in the number of input objects and attributes per generated concept.

## Formal Concept Analysis tools

Many FCA software applications are available today. The main purpose of these tools is to generate the concepts Lattice of a given formal context and the corresponding association rules. These tools are academic and still under active development. One can find a non exhaustive list of FCA tools in the FCA software website. Most of these tools are Java-based open-source application like ConExp, ToscanaJ, Lattice Miner, Coron, etc.

**Chapter 8**

# Formal Ontology

A **formal ontology** is an ontology with a structure that is guided and defined through axioms. The goal of a formal ontology is to provide an unbiased (domain- and application-independent) view on reality. Formal ontologies are founded upon a specific formal upper level ontology, which provides consistency checks for the entire ontology and, if applied properly, allows the modeler to avoid possibly erroneous ontological assumptions encountered in modeling large-scale ontologies.

By maintaining an independent view on reality the ontology gains the following properties:

- indefinite expandability:

  the ontology remains consistent with increasing content.

- content and context independence:

  any kind of 'concept' can find its place.

- accommodate different levels of granularity.

Theories on how to conceptualize reality date back as far as Plato and Aristotle.

## *Existing formal upper-level ontologies (foundational ontologies)*

- BFO - Basic Formal Ontology
- DOLCE - Descriptive Ontology for Linguistic and Cognitive Engineering
- GFO - General Formal Ontology
- OCHRE - Object-Centered High-level REference ontology
- SUMO - Suggested Upper Merged Ontology

- UMBEL - Upper Mapping and Binding Exchange Layer

## *Common terms in formal ontologies*

The Difference in terminology used between separate formal upper level ontologies can be quite substantial, but the one and foremost dichotomy most formal upper level ontologies apply is that between endurants and perdurants.

### Endurant

Also known as continuant, or in some cases 'substance'. Endurants are those entities that can be observed-perceived as a complete concept, at no matter which given snapshot of time. Were we to freeze time we would still be able to perceive/conceive the entire endurant. Examples are material objects, such as an apple or a human, and abstract 'fiat' objects, such as an organisation or the border of a country.

### Perdurant

Also known as occurrent, accident or happening. Perdurants are those entities for which only a part exists if we look at them at any given snapshot in time. When we freeze time we can only see a part of the perdurant. Perdurants are often what we know as processes, for example 'running'. If we freeze time then we only see a part of the running, without any previous knowledge one might not even be able to determine the actual process as being a process of running. Other examples include an activation, a kiss, or a procedure.

### Qualities

In a broad sense, qualities can also be known as properties or tropes. Qualities do not exist on their own, but they need another entity (in many formal ontologies this entity is restricted to be an endurant) in which they occupy. Examples of qualities and the values they assume are colors (red color), or temperatures (warm).

Most formal upper level ontologies recognize qualities, attributes, tropes, or something related, although the exact classification may differ. Some see qualities and the values they can assume (sometimes called quale) as a separate hierarchy besides endurant & perdurant (example: DOLCE). Others classify qualities as a subsection of endurants, e.g. the dependent endurants (example: BFO). Others consider property-instances or tropes that are single characteristics of individuals to be the atoms of the ontology, the simpler entities of which all other entities are composed, so that all the entities are sums or bundles of tropes (example: OCHRE).

## *Formal versus non-formal*

### Example

An ontology might contain a concept representing 'mobility of the arm'. In a nonformal ontology a concept like this can often be classified as for example a 'finding of the arm', right next to other concepts such as 'bruising of the arm'. This method of modeling might create problems with increasing amounts information, as there is no foolproof way to keep hierarchies like this, or their descendant hierarchies (one is a process, the other is a quality) from entangling or knotting.

In a formal ontology, there is an optimal way to properly classify this concept, it is a kind of 'mobility', which is a kind of quality/property (see above). As a quality, it is said to *inhere* in *independent* endurant entities (see above), as such, it cannot exist without a bearer (in the case the arm).

## *Applications for formal ontologies*

### Formal ontology as a template to create novel specific domain ontologies

Having a formal ontology at your disposal, especially when it consists of a Formal upper layer enriched with concrete domain-independent 'middle layer' concepts, can really aid the creation of a domain specific ontology. It allows the modeller to focus on the content of the domain specific ontology without having to worry on the exact higher structure or abstract philosophical framework that gives his ontology a rigid backbone. Disjoint axioms at the higher level will prevent many of the commonly made ontological mistakes made when creating the detailed layer of the ontology.

### Formal ontology as a crossmapping hub: crossmapping taxonomies, databases and non-formal ontologies

Aligning terminologies and ontologies is not an easy task. The divergence of the underlying meaning of word descriptions and terms within different information sources is a well known obstacle for direct approaches to data integration and mapping. One single description may have a completely different meaning in one data source when compared with another. This is because different databases/terminologies often have a different viewpoint on similar items. They are usually built with a specific application-perspective in mind and their hierarchical structure represents this.

A formal ontology, on the other hand, represents entities without a particular application scope. Its hierarchy reflects ontological principles and a basic class-subclass relation between its concepts. A consistent framework like this is ideal for crossmapping data sources. However, one cannot just integrate these external data sources in the formal ontology. A direct incorporation would lead to corruption of the framework and principles of the formal ontology.

A formal ontology is a great crossmapping hub only if a complete distinction between the content and structure of the external information sources and the formal ontology itself is maintained. This is possible by specifying a mapping relation between concepts from a chaotic external information source and a concept in the formal ontology that corresponds with the meaning of the former concept.

Where two or more external information sources map to one and the same formal ontology concept a crossmapping/translation is achieved, as you know that those concepts - no matter what their phrasing is - mean the same thing.

## Formal ontology to empower natural language processing

In ontologies designed to serve natural language processing (NLP) and natural language understanding (NLU) systems ontology concepts are usually connected and symbolized by terms. This connection represents a linguistic realization. Terms are words or a combination of words (multi-word units), in different languages, used to describe in natural language an element from reality, and hence connected to that formal ontology concept that frames this element in reality.
The lexicon, the collection of terms and their inflections assigned to the concepts and relationships in an ontology, forms the 'ontology interface to natural language', the channel through which the ontology can be accessed from a natural language input.

## Formal ontology to normalize database/instance data

The great thing about a formal ontology, in contrast to rigid taxonomies or classifications, is that it allows for indefinite expansion. Given proper modeling, just about any kind of conceptual information, no matter the content, can find its place.

Chapter 9

# Semantic Interoperability

**Semantic Interoperability** is a term used in computer science as a synonym for "Computable Semantic Interoperability". In this sense, it is the ability of computer systems to communicate information and have that information properly interpreted by the receiving system in the same sense as intended by the transmitting system. "Proper interpretation" means that the transmitted information will be used appropriately by a receiving computer system because the logical implications derivable from transmitted information will be the same as those that the sending system would derive. Semantic Interoperability requires that any two systems will derive the same inferences from the same information. This term is sometimes also used as a synonym for "General Semantic Interoperability", the ability of computer systems to place information in a public location and have that information properly interpreted by systems whose developers do not know the creators of the information nor the purpose for which it was created.

**Semantic Interoperability** is also use in a more general sense, as the ability of any communicating entities (not only computers) to share unambiguous meaning. In this broader sense, the **sender** must be able to reliably transmit all sufficient and necessary information; the **receiver** must be able to correctly interpret its interlocutor; and both must be aware of, and agree upon, each other behaviors for given interactions.

## *Semantic as a function of syntactic and pragmatic interoperability*

Syntactic interoperability, provided by for instance XML or the SQL standards, is a pre-requisite to semantic. It involves a common data format and common protocol to structure any data so that the manner of processing the information will be interpretable from the structure. It also allows detection of syntactic errors, thus allowing receiving systems to request resending of any message that appears to be garbled or incomplete. No semantic communication is possible if the syntax is garbled or unable to represent the data. However, information represented in one syntax may in some cases be accurately

translated into a different syntax. Where accurate translation of syntaxes is possible, systems using different syntaxes may also interoperate accurately. In some cases the ability to accurately translate information among systems using different syntaxes may be limited to one direction, when the formalisms used have different levels of *expressivity* (ability to express information).

However, once the syntactical correctness has been verified, the intended meaning of the content of a communication still cannot be judged without *some* commonality in methods and procedures that each system is employing for *semantic* interpretation, which goes beyond the syntactic. In other words, the use of the data or context of application must be understood and unambiguously defined. The task of achieving semantic interoperabiilty among computer systems requires the use of a means to assure that, if there is any context sensitivity to the way terms are used, that the context must also be specified as part of the information using those terms.

Semantic interoperability may be achieved in an interactive manner among a limited group of systems that communicate with each other regularly, so as to allow refinement or clarification of meanings that are unclear, or addition of new meanings. Messages like "this doesn't seem to be an appropriate action at this moment" might provide feedback to the semantic interoperability layer of such interacting systems to suggest it has made errors. Over time, also, state of a system may change or the agreements that govern it may change as more and more systems are aligned. Messages that are semantically unambiguous at one time may be ambiguous if viewed later after the range of possible messages becomes more refined.

Just as computer programs are very often specified both from top-down user requirements or design precedents and bottom-up system capabilities (like shared libraries and APIs), semantics of a local system or group of systems in communication with each other often emerge from compromises between its syntax and its pragmatics.

The more ambitious goal of General Semantic Interoperability presupposes that such interactive refinements or clarifications of meaning are not possible. In that case, the meanings of any information available to multiple remote systems must be specified in sufficient detail to resolve any potential ambiguity. This requires the use of some common standard of meaning. The current best technology for achieving that level of precision in specification of meaning is the use of a logical representation at least as expressive a First-Order Logic (FOL). A common ontology allows all interoperating systems to specify meanings of terms with precision, by linking terms used in specific contexts to the ontology elements that describe the meanings of those terms in logical format.

A single ontology containing representations of every term used in every application is generally considered impossible, because of the rapid creation of new terms or assignments of new meanings to old terms. However, though it is impossible to anticipate *every* concept that a user may wish to represent in a computer, there is the possibility of finding some finite set of "primitive" concept representations that can be combined to

create any of the more specific concepts that users may need for any given set of applications or ontologies. Having a foundation ontology (also called *upper ontology*) that contains all those primitive elements would provide a sound basis for general semantic interoperability, and allow users to define any new terms they need by using the basic inventory of ontology elements, and still have those newly-defined terms properly interpreted by any other computer system that can interpret the basic foundation ontology. Whether the number of such primitive concept representations is in fact finite, or will expand indefinitely, is a question under active investigation. If it is finite, then a stable foundation ontology suitable to support accurate and general semantic interoperability can evolve after some initial foundation ontology has been tested and used by a wide variety of users. At the present time, no foundation ontology has been adopted by a wide community, so such a stable foundation ontology is still in the future.

## *Words and Meanings*

One persistent misunderstanding recurs in discussion of semantics - the confusion of words and meanings. The meanings of words change, sometimes rapidly. But a formal language such as used in an ontology can encode the meanings (semantics) of concepts in a form that does not change. In order to determine what is the meaning of a particular word (or term in a database, for example) it is necessary to label each fixed concept representation in an ontology with the word(s) or term(s) that may refer to that concept. When multiple words refer to the same (fixed) concept, in language this is called synonymy; when one word is used to refer to more than one concept, that is called ambiguity. Ambiguity and synonymy are among the factors that make computer understanding of language very difficult. The use of words to refer to concepts (the meanings of the words used)is very sensitive to the context and the purpose of any use for many human-readable terms. The use of ontologies in supporting semantic interoperability is to provide a fixed set of concepts whose meanings and relations are stable and can be agreed to by users. The task of determining which terms in which contexts (each database is a different context) then is separated from the task of creating the ontology, and must be taken up by the designer of a database, or the designer of a form for data entry, or the developer of a program for language understanding. When a word used in some interoperability context changes its meaning, then to preserve interoperability it is necessary to change the pointer to the ontology element(s) that specifies the meaning of that word.

An example may clarify the point above. An initial ontology used for interoperability may specify that every instance of type 'Automobile' must have four supporting wheels. Later it is learned that automobiles exist or are developed that have only three wheels. Such three-wheeled automobiles will **not** be instances of that initial ontology concept. There are two ways to remedy the problem. The original ontology concept may be changed to include both three or four-wheeled versions, but that could cause errors in legacy systems that depend on the original meaning of the concept. Instead, one would preferably create a parent (more general) type in the ontology that includes both three and four-wheeled vehicles, and the original 'Automobile' type would then be made a subtype of that new type. To avoid name clashes, that new concept may be called, e.g.

"GenericAutomobile". The documentation of the original concept would be changed to alert users that a broader similar concept representation ('GenericAutomobile')is now available that includes three-wheeled vehicles. The linguistic term 'automobile' in any local terminology could be mapped to the more general type (if three-wheeled automobiles were intended), or optionally to the more restricted type having only four wheels, if that local community only wants to talk about four-wheeled automobiles. Making any change in an ontology can cause problems for legacy users; in this case, some users may want the meaning of 'automobile' to include three-wheeled automobiles, but used the original ontology concept because it was the closest concept available. When the new more general concept is added, all users need to be informed of the change so they can determine if they need to change the pointers from their local terms or data elements to the new ontology element.

This scenario illustrates the importance of trying to achieve stability in the ontology used to specify the meanings of terms used in applications, so as to achieve accurate semantic interoperability. For situations where an ontology is developed for use by a closely interacting group of users, direct communication of needs and requirements can assure that the ontology contains representations of all the concepts required by the users. For the general case of semantic interoperability among a wide range of users, who do not have direct contact with each other, the need to anticipate the requirements of users with many different purposes suggests the need to develop some common ontology, in consultation with a broad diversity of potential users, that has the capability of representing any of the concepts those users need, by combining the primitive concept representations. That is the purpose of a Foundation Ontology with a complete set of representations of the semantic primitives, as described in the previous section.

## *Partial semantic interoperability*

To achieve perfect semantic interoperability, all communicating systems must use term (or symbol) definitions that are identical or can be accurately interconverted. Thus a common ontology is the ideal situation for semantic interoperability. Where that is impossible, lesser degrees of semantic interoperability may be achieved by techniques that automatically map the definitions used by one system to those of another.

Interoperability is sometimes considered as an all-or-nothing attribute of computer systems - *see upper ontology* - but for complex information, different levels of interoperability can be envisioned; when multiple pieces of information are being transferred, correct interpretation of some fraction of that information may be considered as constituting some level of semantic interoperability. Perfect semantic interoperability would require the correct interpretation of all transferred information, from the point of view of all users.

## *Knowledge representation requirements and languages*

A knowledge representation language may be sufficiently expressive to describe nuances of meaning in well understood fields. There are at least five levels of complexity of these.

For general semi-structured data one may use a general purpose language such as XML.

For structured data with well understood relationships one may use SQL or the relational model.

A description logic (such as the one used in the OWL semantic web ontology language) is more complex.

Languages with the full power of first-order predicate logic may be required for many tasks.

Human languages are highly expressive, but are considered too ambiguous to allow the accurate interpretation desired, given the current level of human language technology. In human languages the same word may be used to refer to different concepts (ambiguity), and the same concept may be referred to by different words (synonymy).

## *Prior agreement not required*

Semantic interoperability may be distinguished from other forms of interoperability by considering whether the information transferred has, in its communicated form, all of the meaning required for the receiving system to interpret it correctly, even when the algorithms used by the receiving system are unknown to the sending system. Consider sending one number:

If that number is intended to be the sum of money owed by one company to another, it implies some action or lack of action on the part of both those who send it and those who receive it.

It may be correctly interpreted if sent in response to a specific request, and received at the time and in the form expected. This correct interpretation does not depend only on the number itself, which could represent almost any of millions of types of quantitative measure, rather it depends strictly on the circumstances of transmission. That is, the interpretation depends on both systems expecting that the algorithms in the other system use the number in exactly the same sense, and it depends further on the entire envelope of transmissions that preceded the actual transmission of the bare number. By contrast, if the transmitting system does not know how the information will be used by other systems, it is necessary to have a shared agreement on how information with some specific meaning (out of many possible meanings) will appear in a communication. For a particular task, one solution is to standardize a form, such as a request for payment; that request would have to encode, in standardized fashion, all of the information needed to evaluate it, such as: the agent owing the money, the agent owed the money, the nature of the action giving rise to the debt, the agents, goods, services, and other participants in that action; the time of the action; the amount owed and currency in which the debt is reckoned; the time allowed for payment; the form of payment demanded; and other information. When two or more systems have agreed on how to interpret the information in such a request, they can achieve semantic interoperability *for that specific type of transaction*. For semantic

interoperability generally, it is necessary to provide standardized ways to describe the meanings of many more things than just commercial transactions, and the number of concepts whose representation needs to be agreed upon are at a minimum several thousand.

## *Ontology research*

How to achieve semantic interoperability for more than a few restricted scenarios is currently a matter of research and discussion. For the problem of General Semantic Interoperability, some form of foundation ontology ('upper ontology') is required that is sufficiently comprehensive to provide the defining concepts for more specialized ontologies in multiple domains. Over the past decade more than ten foundation ontologies have been developed, but none have as yet been adopted by a wide user base.

The need for a single comprehensive all-inclusive ontology to support Semantic Interoperability can be avoided by designing the common foundation ontology as a set of basic ("primitive") concepts that can be combined to create the logical descriptions of the meanings of terms used in local domain ontologies or local databases. This tactic is based on the principle that:

**If:**

```
(1) the meanings and usage of the primitive ontology elements in the
foundation ontology are agreed on, and
(2) the ontology elements in the  domain ontologies are constructed as
logical
combinations of the elements in the foundation ontology,
```

**Then:**

```
The intended meanings of the domain ontology elements can be computed
automatically using an FOL reasoner, by any system that accepts the
meanings of the elements in the foundation ontology, and has both the
foundation ontology and the logical specifications of the elements in
the domain ontology.
```

**Therefore:**

```
Any system wishing to interoperate accurately with another system need
transmit only the data to be communicated, plus any logical
descriptions of terms used in that data that were created locally and
are not already in the common foundation ontology.
```

This tactic then limits the need for prior agreement on meanings to only those ontology elements in the common Foundation Ontology (FO). Based on several considerations, this is likely to be fewer than 10,000 elements (types and relations).

In practice, together with the FO focused on representations of the primitive concepts, a set of domain extension ontologies to the FO with elements specified using the FO

elements will likely also be used. Such pre-existing extensions will ease the cost of creating domain ontologies by providing existing elements with the intended meaning, and will reduce the chance of error by using elements that have already been tested. Domain extension ontologies may be logically inconsistent with each other, and that needs to be determined if different domain extensions are used in any communication.

Whether use of such a single foundation ontology can itself be avoided by sophisticated mapping techniques among independently developed ontologies is also under investigation.

## *Importance*

The practical significance of semantic interoperability has been measured by several studies that estimate the cost (in lost efficiency) due to lack of semantic interoperability. One study, focusing on the lost efficiency in the communication of healthcare information, estimated that US$77.8 billion per year could be saved by implementing an effective interoperability standard in that area. Other studies, of the construction industry and of the automobile manufacturing supply chain, estimate costs of over US$10 billion per year due to lack of semantic interoperability in those industries. In total these numbers can be extrapolated to indicate that well over US$100 billion per year is lost because of the lack of a widely used semantic interoperability standard in the US alone.

There has not yet been a study about each policy field that might offer big cost savings applying semantic interoparability standards. But to see which policy fields are capable of profiting from semantic interoperability see 'Interoperability' in general. Such policy fields are eGovernment, health, security and many more. The EU also set up the Semantic Interoperability Centre Europe in June 2007.

Chapter 10

# Process Ontology

A **process ontology** is a description of the components and their relationships that make up a process. A formal process ontology is an ontology in the knowledge domain of processes. Often such ontologies take advantage of the benefits of an upper ontology. Planning software can be used to perform plan generation based on the formal description of the process and its constraints. Numerous efforts have been made to define a process/planning ontology.

## *Processes*

A process may be defined as a set of transformations of input elements into output elements with specific properties, with the transformations characterized by parameters and constraints, such as in manufacturing or biology. A process may also be defined as the workflows inherent in processes such as manufacturing, engineering and business processes.

## *Ontologies*

### PSL

The Process Specification Language (PSL) is a process ontology developed for the formal description and modeling of basic manufacturing, engineering and business processes. This ontology provides a vocabulary of classes and relations for concepts at the ground level of event-instances, object-instances, and timepoints. PSL's top level is built around the following:

- Activity, a class or type of action, such as install-part, which is the class of actions in which parts are installed
- Activity-occurrence, an event or action that takes place at a specific place and time, such as a specific instance of install-part occurring at a specific timestamp
- Timepoint, a point in time
- Object, anything that is not a timepoint or an activity

## Cyc

In a process/planning ontology developed for the ontology Cyc, classes and relations above the ground level of PSL allow processes to be described purely at the type-level. The ground level of PSL uses the primitives of event-instance, object-instance, and timepoint description. The types above this ground level are the concepts of scripts/scenes, roles/participants, conditions/constraints, and repetition. Identity of a participant in a process is represented by conditions/constraints. These types have also been expressed in PSL, showing that the type level and the ground level are relatively independent.

## SUPER and DDPO

The project SUPER (**S**emantics **U**tilised for **P**rocess management within and between **E**nte**R**prises) has a goal of the definition of ontologies for Semantic Business Process Management (SBPM), but these ontologies can be reused in diverse environments. Part of this project is to define an Upper Process Ontology (UPO) that ties together all other SUPER ontologies. The results of the project SUPER include the UPO and a set of ontologies for processes and organizations. Most of the ontologies are written in WSML, and some are also written in OCML.

A candidate model for the UPO was DDPO (DOLCE+DnS Plan Ontology), a planning ontology which specifies plans and distinguishes between abstract and executable plans. DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) aims at capturing the ontological categories underlying natural language and human commonsense. DnS (Descriptions and Situations), is a constructivist ontology that allows for context-sensitive redescriptions of the types and relations postulated by other given ontologies (or ground vocabularies). Together in DDPO, DOLCE and DnS are used to build a Plan Ontology that includes physical and non-physical objects (social entities, mental objects and states, conceptualizations, information objects, constraints), events, states, regions, qualities, and constructivist situations. The main target of DDPO is tasks, namely the types of actions, their sequencing, and the controls performed on them.

## oXPDL

The ontology oXPDL is a process interchange ontology based on the standardised XML Process Definition Language (XPDL). The purpose of oXPDL is to model the semantics of XPDL process models in standardized Web ontology languages such as OWL and WSML, while incorporating features of existing standard ontologies such as PSL, RosettaNet, SUMO, and eClassOWL.

## GFO

The General Formal Ontology (GFO) is an ontology integrating processes and objects. GFO includes elaborations of categories like objects, processes, time and space, properties, relations, roles, functions, facts, and situations. GFO allows for different

axiomatizations of its categories, such as the existence of atomic time-intervals vs. dense time. Two of the specialties of GFO are its account of persistence and its time model. Regarding persistence, the distinction between endurants (objects) and perdurants (processes) is made explicit within GFO by the introduction of a special category, a persistant. A persistant is a special category with the intention that its instances "remain identical" over time. With respect to time, time intervals are taken as primitive in GFO, and time-points (called "time boundaries") are derived. Moreover, time-points may coincide, which is convenient for modelling instantaneous changes.

## m3po and m3pl

The multi metamodel process ontology (m3po) combines workflows and choreography descriptions so that it can be used as a process interchange ontology. For internal business processes, Workflow Management Systems are used for process modelling and allow describing and executing business processes. For external business processes, choreography descriptions are used to describe how business partners can cooperate. A choreography can be considered to be a view of an internal business process with the internal logic not visible, similar to public views on private workflows. The m3po ontology unifies both internal and external business processes, combining reference models and languages from the workflow and choreography domains. The m3po ontology is written in WSML. The related ontology m3pl, written in PSL using the extension FLOWS (First Order Logic for Web Services), enables the extraction of choreography interfaces from workflow models.

The m3po ontology combines features of the following reference models and languages:

- XPDL - a standard for exchanging workflow models, without runtime information, between different workflow management systems
- PSL - an ontology that allows the capture of the semantics of workflow models and enables translations of models between workflow management systems
- YAWL - a research workflow language that supports all workflow patterns directly
- BPEL - an executable business process language and includes an abstract protocol
- WS-CDL - a multi-party collaboration model

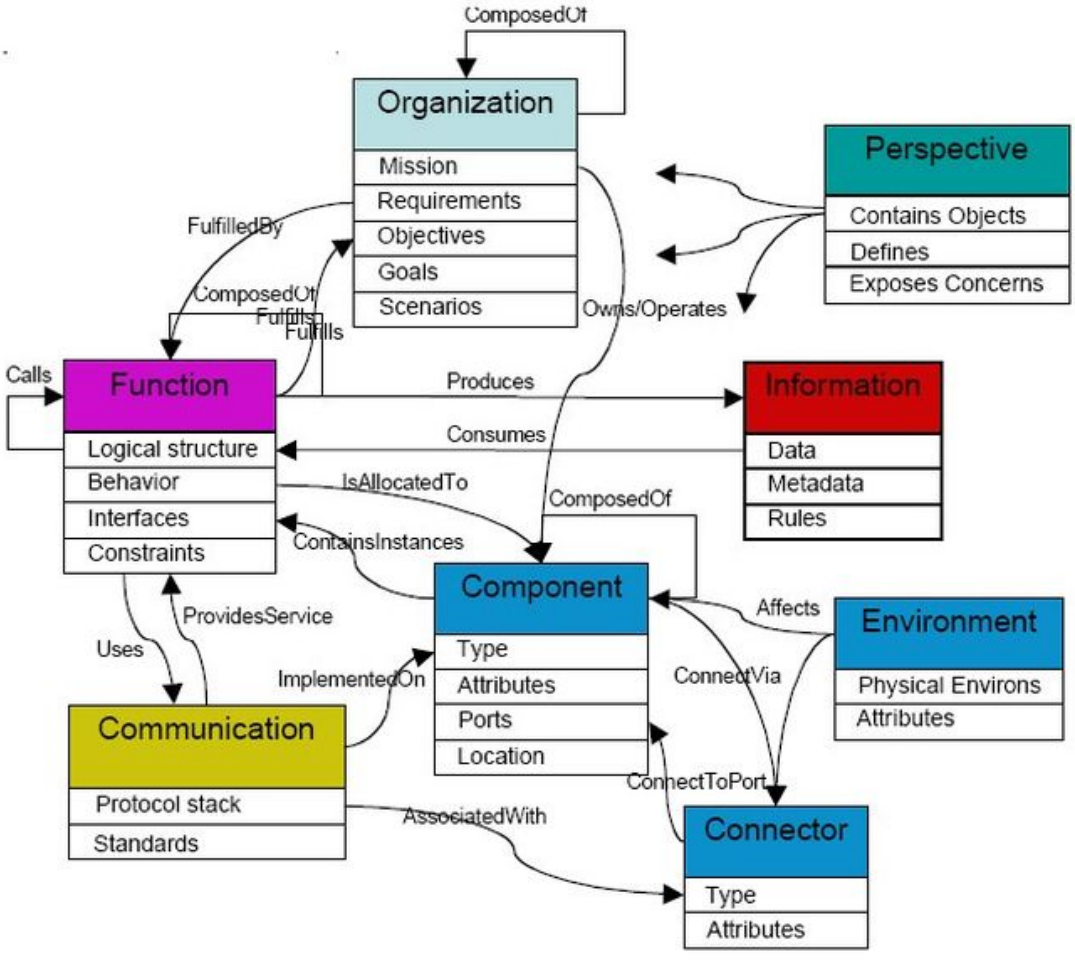
The m3po ontology is organized using five key aspects of workflow specifications and workflow management. Because different workflow models put a different emphasis on the five aspects, the most elaborate reference model for each aspect was used and combined into m3po.

- Functional and Behavioural - the most important concepts are *processType*, *processOccurrence*, *activityType* and *activityOccurrence*
- Informational - defined by data and data-flow
- Organizational - defines who is responsible for carrying out a specific task; security related issues

- Operational - interaction of the workflows with their environment by manual tasks performed by users and automatic tasks performed by automated computer programs
- Orthogonal - scheduling based on time; integrity and failure recovery

**Chapter 11**

# Ontology Engineering




Example of a constructed MBED Top Level Ontology based on the Nominal set of views.

**Ontology engineering** in computer science and information science is a new field, which studies the methods and methodologies for building ontologies: formal representations of a set of concepts within a domain and the relationships between those concepts. A large scale representation of abstract concepts such as actions, time, physical objects and beliefs would be an example of ontological engineering.

## *Overview*

[Ontology engineering] aims at making explicit the knowledge contained within software applications, and within enterprises and business procedures for a particular domain. Ontology engineering offers a direction towards solving the inter-operability problems brought about by semantic obstacles, i.e. the obstacles related to the definitions of business terms and software classes. Ontology engineering is a set of tasks related to the development of ontologies for a particular domain.

— Line Pouchard, Nenad Ivezic and Craig Schlenoff, *Ontology Engineering for Distributed Collaboration in Manufacturing*

Ontologies provide a common vocabulary of an area and define, with different levels of formality, the meaning of the terms and the relationships between them. During the last decade, increasing attention has been focused on ontologies. Ontologies are now widely used in knowledge engineering, artificial intelligence and computer science; in applications related to areas such as knowledge management, natural language processing, e-commerce, intelligent information integration, bio-informatics, education; and in new emerging fields like the semantic web. Ontological engineering is a new field of study concerning the ontology development process, the ontology life cycle, the methods and methodologies for building ontologies, and the tool suites and languages that support them.

## *Ontology languages*

An ontology language is a formal language used to encode the ontology. There are a number of such languages for ontologies, both proprietary and standards-based:

- Common logic is ISO standard 24707, a specification for a family of ontology languages that can be accurately translated into each other.
- The Cyc project has its own ontology language called CycL, based on first-order predicate calculus with some higher-order extensions.
- The Gellish language includes rules for its own extension and thus integrates an ontology with an ontology language.
- IDEF5 is a software engineering method to develop and maintain usable, accurate, domain ontologies.
- KIF is a syntax for first-order logic that is based on S-expressions.
- Rule Interchange Format (RIF) and F-Logic combine ontologies and rules.
- OWL is a language for making ontological statements, developed as a follow-on from RDF and RDFS, as well as earlier ontology language projects including

OIL, DAML and DAML+OIL. OWL is intended to be used over the World Wide Web, and all its elements (classes, properties and individuals) are defined as RDF resources, and identified by URIs.

- XBRL (Extensible Business Reporting Language) is a syntax for expressing business semantics.

## *Ontology Engineering In Life Sciences*

Life sciences is flourishing with ontologies that biologists use to make sense of their experiments. For inferring correct conclusions from experiments, ontologies have to be structured optimally against the knowledge base they represent. The structure of an ontology needs to be changed continuously so that it is an accurate representation of the underlying domain.

Recently, an automated method was introduced for engineering ontologies in life sciences such as Gene Ontology (GO), one of the most successful and widely used biomedical ontology. Based on information theory, it restructures ontologies so that the levels represent the desired specificity of the concepts. Similar information theoretic approaches have also been used for optimal partition of Gene Ontology. Given the mathematical nature of such engineering algorithms, these optimizations can be automated to produce a principled and scalable architecture to restructure ontologies such as GO.

## *Tools for ontology engineering*

- DOGMA
- DogmaModeler
- KAON
- OntoClean
- OnToContent
- HOZO
- Protégé (software)

Chapter 12

# Upper Ontology (Information Science)

In information science, an **upper ontology** (**top-level ontology**, or **foundation ontology**) is an ontology which describes very general concepts that are the same across all knowledge domains. The most important function of an upper ontology is to support very broad semantic interoperability between a large number of ontologies accessible "under" this upper ontology. As the metaphor suggests, it is usually a hierarchy of entities and associated rules (both theorems and regulations) that attempts to describe those general entities that do not belong to a specific problem domain.

The seemingly conflicting use of metaphors implying a solid rigorous bottom-up "foundation" or a top-down imposition of somewhat arbitrary and possibly political decisions is no accident - the field is characterized by controversy, politics, competing approaches and academic rivalry.

Debates notwithstanding, it can be said that a very important part of each upper ontology can be considered as the computational implementation of natural philosophy, which itself is a more empirical method for investigating the topics within the philosophical discipline of physical ontology.

Library classification systems predate these upper ontology systems. Though library classifications organize and categorize knowledge using general concepts that are the same across all knowledge domains, neither system is a replacement for the other.

## *Development*

Upper ontologies are also commercially valuable, creating competition to define them. Peter Murray-Rust has claimed that this leads to "semantic and **ontological warfare** due to competing standards" , and accordingly any standard foundation ontology is likely to be contested among commercial or political parties, each with their own idea of 'what exists'.. An important factor exacerbating the failure to arrive at a common approach is

the absence of open-source applications that would permit the testing of different ontologies in the same computational environment. The differences are debated largely on theoretical grounds, or are merely the result of personal preferences, with no method to objectively compare practical performance.

No one upper ontology has yet gained widespread acceptance as a de facto standard. Different organizations are attempting to define standards for specific domains. The 'Process Specification Language' (PSL) created by the National Institute for Standards and Technology (NIST) is one example.

An important factor leading to the absence of wide adoption of any existing upper ontology is the complexity. An upper ontology typically has from 2000 to 10,000 elements (classes, relations), with complex interactions among them. The resulting complexity is similar to that of a human language, and the learning process can be even longer than for a human language because of the unfamiliar format and logical rules. The motivation to overcome this learning barrier is largely absent because of the paucity of publicly accessible examples of use. As a result, those building domain ontologies for a local application tend to create the simplest possible domain-specific ontology, not related to any upper ontology. Such domain ontologies may function adequately for the local purpose, but are very time-consuming to relate accurately to other domain ontologies.

There is debate over whether the concept of using a single, shared upper ontology is even feasible or practical at all. There is further debate over whether the debates are valid - often leading to outright censorship and boosterism of particular approaches in supposedly neutral sources *including this one*. Some of these arguments are outlined below, with no attempt to be comprehensive. Please do not censor them because you promote some ontology.

## Arguments for the infeasibility of an upper ontology

Historically, many attempts in many societies have been made to impose or define a single set of concepts as more primal, basic, foundational, authoritative, true or rational than others.

In the kind of modern societies that have computers at all, the existence of academic and political freedoms imply that many ontologies will simultaneously exist and compete for adherents. While the differences between them may be narrow and appear petty to those not deeply involved in the process, so too did many of the theological debates of medieval Europe, but they still led to schisms or wars, or were used as excuses for same. The tyranny of small differences that standard ontologies seek to end may continue simply because other forms of tyranny are even less desirable. So private efforts to create competitive ontologies that achieve adherents by virtue of better communication may proceed, but tend not to result in long standing monopolies.

A deeper objection derives from ontological constraints that philosophers have found historically inescapable. Some argue that a transcendent perspective or omniscience is implied by even searching for any general purpose ontology - *see God's eye view*- since it is a social / cultural artifact, there is no purely objective perspective from which to observe the whole terrain of concepts and derive any one standard.

A narrower and much more widely held objection is implicature: the more general the concept and the more useful in semantic interoperability, the less likely it is to be reducible to symbolic concepts or logic and the more likely it is to be simply accepted by the complex beings and cultures relying on it. In the same sense that a fish doesn't perceive water, we don't see how complex and involved is the process of understanding basic concepts.

- There is no self-evident way of dividing the world up into concepts, and certainly no non-controversial one
- There is no neutral ground that can serve as a means of translating between specialized (or "lower" or "application-specific") ontologies
- Human language itself is already an arbitrary approximation of just one among many possible conceptual maps. To draw any *necessary correlation* between English words and any number of intellectual concepts we might like to represent in our ontologies is just asking for trouble. (WordNet, for instance, is successful and useful precisely because it does not pretend to be a general-purpose upper ontology; rather, it is a tool for semantic / syntactic / linguistic disambiguation, which is richly embedded in the particulars and peculiarities of the English language.)
- Any hierarchical or topological representation of concepts must begin from some ontological, epistemological, linguistic, cultural, and ultimately pragmatic perspective. Such pragmatism does not allow for the exclusion of politics between persons or groups, indeed it requires they be considered as perhaps more basic primitives than any that are represented.

Those who doubt the feasibility of general purpose ontologies are more inclined to ask “what specific purpose do we have in mind for this conceptual map of entities and what practical difference will this ontology make?” This pragmatic philosophical position surrenders all hope of devising the encoded ontology version of “everything that is the case,” (Wittgenstein, Tractatus Logico-Philosophicus).

According to Barry Smith in *The Blackwell Guide to the Philosophy of Computing and Information* (2004), "the initial project of building one single ontology, even one single top-level ontology, which would be at the same time non-trivial and also readily adopted by a broad population of different information systems communities, has largely been abandoned." (p. 159)

Finally there are objections similar to those against artificial intelligence. Technically, the complex concept acquisition and the social / linguistic interactions of human beings suggests any axiomatic foundation of "most basic" concepts must be cognitive, biological

or otherwise difficult to characterize since we don't have axioms for such systems. Ethically, any general-purpose ontology could quickly become an actual tyranny by recruiting adherents into a political program designed to propagate it and its funding means, and possibly defend it by violence. Historically, inconsistent and irrational belief systems have proven capable of commanding obedience to the detriment of harm of persons both inside and outside a society that accepts them. How much more harmful would a consistent rational one be, were it to contain even one or two basic assumptions incompatible with human life?

## Arguments for the feasibility of an upper ontology

Many of those who doubt the possibility of developing wide agreement on a common upper ontology fall into one of two traps: (1) they assert that there is no possibility of universal agreement on any conceptual scheme; **but** they ignore the fact that a practical common ontology does not need to have universal agreement, it only needs a large enough user community to make it profitable for developers to use it as a means to general interoperability, and for third-party developer to develop utilities to make it easier to use; and (2) they point out that developers of data schemes find different representations congenial for their local purposes; **but** they do not demonstrate that these different representation are in fact logically inconsistent. In fact, different representations of assertions about the real world (though not philosophical models), if they accurately reflect the world, must be logically consistent, even if they focus on different aspects of the same physical object or phenomenon. If any two assertions about the real world are logically inconsistent, one or both must be wrong, and that is a topic for experimental investigation, not for ontological representation. In practice, representations of the real world are created as and known to be approximations to the basic reality, and their use is circumscribed by the limits of error of measurements in any given practical application. Ontologies are entirely capable of representing approximations, and are also capable of representing situations in which different approximations have different utility. Objections based on the different ways people perceive things attack a simplistic, impoverished view of ontology. The objection that there are logically incompatible models of the world are true, but in an upper ontology those different models can be represented as different theories, and the adherents of those theories can use them in preference to other theories, while preserving the logical consistency of the *necessary* assumptions of the upper ontology. The *necessary* assumptions provide the logical vocabulary with which to specify the meanings of all of the incompatible models. It has never been demonstrated that incompatible models cannot be properly specified with a common, more basic set of concepts, while there are examples of incompatible theories that can be logically specified with only a few basic concepts.

Many of the objections to upper ontology refer to the problems of life-critical decisions or non-axiomatized problem areas such as law or medicine or politics that are difficult even for humans to understand. Some of these objections do not apply to physical objects or standard abstractions that are defined into existence by human beings and closely controlled by them for mutual good, such as standards for electrical power system connections or the signals used in traffic lights. No single general metaphysics is required

to agree that some such standards are desirable. For instance, while time and space can be represented many ways, some of these are already used in interoperable artifacts like maps or schedules.

Objections to the feasibility of a common upper ontology also do not take into account the possibility of forging agreement on an ontology that contains all of the **primitive** ontology elements that can be combined to create any number of more specialized concept representations. Adopting this tactic permits effort to be focused on agreement only on a limited number of ontology elements (under 10,000). By agreeing on the meanings of that inventory of basic concepts, it becomes possible to create and then accurately and automatically interpret an infinite number of concept representations as combinations of the basic ontology elements. Any domain ontology or database that uses the elements of such an upper ontology to specify the meanings of its terms will be automatically and accurately interoperable with other ontologies that use the upper ontology, even though they may each separately define a large number of domain elements not defined in other ontologies. In such a case, proper interpretation will require that the logical descriptions of domain-specific elements be transmitted along with any data that is communicated; the data will then be automatically interpretable because the domain element descriptions, based on the upper ontology, will be properly interpretable by any system that can properly use the upper ontology. An upper ontology based on such a set of primitive elements can include alternative views, provided that they are logically compatible. Logically incompatible models can be represented as alternative theories, or represented in a specialized extension to the upper ontology. The proper use of alternative theories is a piece of knowledge that can itself be represented in an ontology.

Most proponents of an upper ontology argue that several good ones may be created with perhaps different emphasis. Very few are actually arguing to discover just one within natural language or even an academic field. Most are simply standardizing some existing communication. Another view advanced is that there is almost total overlap of the different ways that upper ontologies have been formalized, in the sense that different ontologies focus on a different aspect of the same entities, but the different views are complementary and not contradictory to each other; as a result, an internally consistent ontology that contains all the views, with means of translating the different views into the other, is feasible. Such an ontology has not thus far been constructed, however, because it would require a large project to develop so as to include all of the alternative views in the separately developed upper ontologies, along with their translations. The main barrier to construction of such an ontology is not the technical issues, but the reluctance of funding agencies to provide the funds for a large enough consortium of developers and users.

Several common arguments against upper ontology can be examined more clearly by separating issues of concept definition (ontology), language (lexicons), and facts (knowledge). For instance, people have different terms and phrases for the same concept. However, that does not necessarily mean that those people are referring to different concepts. They may simply be using different language or idiom. Formal ontologies typically use linguistic labels to refer to concepts, but the terms that label ontology

elements mean no more and no less than what their axioms say they mean. Labels are similar to variable names in software, evocative rather than definitive. The proponents of a common upper ontology point out that the meanings of the elements (classes, relations, rules) in an ontology depend only on their logical form, and not on the labels, which are usually chosen merely to make the ontologies more easily usable by their human developers. In fact, the labels for elements in an ontology need not be words - they could be, for example, images of instances of a particular type, or videos of an action that is represented by a particular type. It cannot be emphasized too strongly that words are *not* what are represented in an ontology, but entities in the real world, or abstract entities (concepts) in the minds of people. Words are not equivalent to ontology elements, but words *label* ontology elements. There can be many words that label a single concept, even in a single language (synonymy), and there can be many concepts labeled by a single word (ambiguity). Creating the mappings between human language and the elements of an ontology is the province of Natural Language Understanding. But the ontology itself stands independently as a logical and computational structure. For this reason, finding agreement on the structure of an ontology is actually easier than developing a controlled vocabulary, because all different interpretations of a word can be included, each *mapped* to the same word in the different terminologies.

A second argument is that people believe different things, and therefore can't have the same ontology. However, people can assign different truth values to a particular assertion while accepting the validity of certain underlying claims, facts, or way of expressing an argument with which they disagree. (Using, for instance, the issue/position/argument form.) This objection to upper ontologies ignores the fact that a single ontology can represent different belief systems, representing them as different belief systems, without taking a position on the validity of either.

Even arguments about the existence of a thing require a certain sharing of a concept, even though its existence in the real world may be disputed. Separating belief from naming and definition also helps to clarify this issue, and show how concepts can be held in common, even in the face of differing belief. For instance, Ency as a medium may permit such confusion but disciplined users can apply dispute resolution methods to sort out their conflicts. It is also argued that most people share a common set of "semantic primitives", fundamental concepts, to which they refer when they are trying to explain unfamiliar terms to other people. An ontology that includes representations of those semantic primitives could in such a case be used to create logical descriptions of any term that a person may wish to define logically. That ontology would be one form of upper ontology, serving as a logical "interlingua" that can translate ideas in one terminology to its logical equivalent in another terminology.

Advocates argue that most disagreement about the viability of an upper ontology can be traced to the conflation of ontology, language and knowledge, or too-specialized areas of knowledge: many people, or agents or groups will have areas of their respective internal ontologies that do not overlap. If they can cooperate and share a conceptual map at all, this may be so very useful that it outweighs any disadvantages that accrue from sharing. To the degree it becomes harder to share concepts the deeper one probes, the more

valuable such sharing tends to get. If the problem is as basic as opponents of upper ontologies claim, then, it applies also to a group of humans trying to cooperate, who might need machine assistance to communicate easily.

If nothing else, such ontologies are implied by machine translation, used when people cannot practically communicate. Whether "upper" or not, these seem likely to proliferate.

## *Available ontologies*

### Cyc

A well-known and quite comprehensive ontology available today is Cyc, a proprietary system under development since 1986, consisting of a foundation ontology and several domain-specific ontologies (called *microtheories*). A subset of that ontology has been released for free under the name OpenCyc, and a more or less unabridged version is made available for non-commercial use under the name ResearchCyc.

### UMBEL

UMBEL is an ontology of 28,000 reference concepts that maps to a simplified subset of the OpenCyc ontology, that is intended to provide a way of linking the precise OpenCyc ontology with less formal ontologies.

### Basic Formal Ontology (BFO)

The BFO or Basic Formal Ontology framework developed by Barry Smith and his associates consists in a series of sub-ontologies at different levels of granularity. The ontologies are divided into two varieties: **SNAP** (or snapshot) ontologies, comprehending continuant entities such as three-dimensional enduring objects, and **SPAN** ontologies, comprehending processes conceived as extended through (or as spanning) time. BFO thus incorporates both three-dimensionalist and four-dimensionalist perspectives on reality within a single framework. Interrelations are defined between the two types of ontologies in a way which gives BFO the facility to deal with both static/spatial and dynamic/temporal features of reality. Each SNAP ontology is an inventory of all entities existing at a time. Each SPAN ontology is an inventory (processory) of all the processes unfolding through a given interval of time. Both types of ontology serve as basis for a series of sub-ontologies, each of which can be conceived as a window on a certain portion of reality at a given level of granularity. An example of an application of BFO can be seen in the Ontology for Biomedical Investigations (OBI). A list of the large number of ontologies based on BFO can be found here.

### DOLCE and DnS

Developed by Nicola Guarino and his associates at the Laboratory for Applied Ontology (LOA), the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) is the first module of the WonderWeb foundational ontologies library. As implied by its

acronym, DOLCE has a clear *cognitive bias*, in that it aims at capturing the ontological categories underlying natural language and human commonsense. DOLCE, however, does not commit to a strictly referentialist metaphysics related to the intrinsic nature of the world. Rather, the categories it introduces are thought of as cognitive artifacts, which are ultimately depending on human perception, cultural imprints and social conventions. In this sense, they intend to be just *descriptive* (vs *prescriptive*) notions, that assist in making already formed conceptualizations explicit. DOLCE is an ontology of particulars, in the sense that its domain of discourse is restricted to them. Of course, universals are used to organize and characterize the particulars, but they are not themselves subject to being organized and characterized (e.g., by means of metaproperties).

DnS (Descriptions and Situations), developed by Aldo Gangemi (STLab, Rome), is a *constructivist* ontology that pushes DOLCE's descriptive stance even further. DnS does not put restrictions on the type of entities and relations that one may want to postulate, either as a domain specification, or as an upper ontology, and it allows for context-sensitive '*redescriptions*' of the types and relations postulated by other given ontologies (or 'ground' vocabularies). The current OWL encoding of DnS assumes DOLCE as a ground top-level vocabulary. DnS and related modules also exploit 'CPs' (Content ontology design Patterns), which provide a framework to annotate 'focused' fragments of a reference ontology (i.e., the parts of an ontology containing the types and relations that underlie 'expert reasoning' in given fields or communities). The combination of DOLCE and DnS has been used to build a planning ontology known as DDPO (DOLCE+DnS Plan Ontology).

Both DOLCE and DnS are particularly devoted to the treatment of social entities, such as e.g. organizations, collectives, plans, norms, and information objects. The DOLCE-2.1-Lite-Plus OWL version, including a number of DnS-based modules, has been and is being applied to several ontology projects.

A lighter OWL axiomatization of DOLCE and DnS, which also simplifies the names of many classes and properties, adds extensive inline comments, and thoroughly aligns to the repository of Content patterns is now available as DOLCE+DnS-Ultralite (abbreviated: DUL). Despite its simplification, which greatly speeds up consistency checking and classification of OWL domain ontologies that are plugged to it, the expressivity of DUL is not significantly different from the previous DOLCE-Lite-Plus. DOLCE OWL versions, DOLCE+DnS-Ultralite and the pattern repository are developed and maintained by Aldo Gangemi and his associates at Rome's Semantic Technology Lab.

## General Formal Ontology (GFO)

The General Formal Ontology (GFO), developed by Heinrich Herre and his colleagues of the research group Onto-Med in Leipzig, is a realistic ontology integrating processes and objects. It attempts to include many aspects of recent philosophy, which is reflected both in its taxonomic tree and its axiomatizations. GFO allows for different axiomatizations of its categories (such as the existence of atomic time-intervals vs. dense time). The basic

principles of GFO are published in the Onto-Med Report Nr. 8 and in General Formal Ontology (GFO): A Foundational Ontology for Conceptual Modelling.

Two GFO specialties, among others, are its account of persistence and its time model. Regarding persistence, the distinction between endurants (objects) and perdurants (processes) is made explicit within GFO by the introduction of a special category, a persistant. A persistant is a special category with the intention that its instances "remain identical" (over time). With respect to time, time intervals are taken as primitive in GFO, and time-points (called "time boundaries") as derived. Moreover, time-points may coincide, which is convenient for modelling instantaneous changes.

## IDEAS

The upper ontology developed by the IDEAS Group is higher-order, extensional and 4D. It was developed using the BORO Method. The IDEAS ontology is not intended for reasoning and inference purposes; its purpose is to be a precise model of business.

## WordNet

WordNet, a freely available database originally designed as a semantic network based on psycholinguistic principles, was expanded by addition of definitions and is now also viewed as a dictionary. It qualifies as an upper ontology by including the most general concepts as well as more specialized concepts, related to each other not only by the subsumption relations, but by other semantic relations as well, such as part-of and cause. However, unlike Cyc, it has not been formally axiomatized so as to make the logical relations between the concepts precise. It has been widely used in Natural language processing research.

## Suggested Upper Merged Ontology

The Suggested Upper Merged Ontology ("SUMO") is another comprehensive ontology project. It includes an upper ontology, created by the IEEE working group P1600.1 (predominantly by Ian Niles and Adam Pease). It is extended with many domain ontologies and a complete set of links to WordNet. It is freely available.

## Biomedical ontology

Examples of *domain ontologies* can be found at the Open Biomedical Ontology site. They act as an umbrella organisation for many ontologies specific to biological topics (such as cellular organelles).

- Open Biomedical Ontologies
- Search, browse and visualise the OBO ontologies online via the NCBO Bioportal
- Ontology browser for most of the Open Biomedical Ontologies

## COSMO

COSMO (COmmon Semantic MOdel) is an ontology that was initiated as a project of the COSMO working group of the Ontology and taxonomy Coordinating Working Group. The current version is an OWL ontology, but a Common-Logic compliant version is anticipated in the future. The ontology and explanatory files are available at the COSMO site. The goal of the COSMO working group was to develop a foundation ontology by a collaborative process that will allow it to represent all of the basic ontology elements that all members feel are needed for their applications. The development of COSMO is fully open, and any comments or suggestions from any sources are welcome. After some discussion and input from members in 2006, the development of the COSMO has been continued primarily by Patrick Cassidy, the chairman of the COSMO Working Group. Contributions and suggestions from any interested party are still welcome and encouraged. Many of the types (OWL classes) in the current COSMO have been taken from the OpenCyc OWL version 0.78, and from the SUMO. Other elements were taken from other ontologies (such as BFO and DOLCE), or developed specifically for COSMO. Recent development of the COSMO has focused on including representations of all of the words in the Longman Dictionary of Contemporary English (LDOCE) controlled defining vocabulary (2148 words). These words are sufficient to define (linguistically) all of the entries in the LDOCE. It is hypothesized that the ontological representations of the concepts represented by those terms will be sufficient to specify the meanings of any specialized ontology element, thereby serving as a basis for general Semantic Interoperability. The current (May 2009) OWL version of COSMO has over 6400 types (OWL classes), over 700 relations, and over 1400 restrictions.

## OCHRE

The Object-Centered High-level REference ontology (OCHRE) was developed by Luc Schneider at the Institute for Formal Ontology and Medical Information Science at the University of Leipzig. This ontology was developed not only to create a particular basic ontological framework, but also to demonstrate how the quality of a foundational ontology depends on descriptive adequacy and on formal simplicity and transparency. The ontology identifies objects, attributes, and events as describing reality. The ontology distinguishes thin objects (a core of enduring characteristics) and thick objects (having spatio-temporal bulk that undergo change; being stages or phases or snapshots of thin objects). Attributes (properties and relations) can be regarded either as repeatables (universals that apply to more than one case) or as non-repeatables (property-instances or tropes that are single characteristics of individuals). Events (changes or state-transitions) are accounted for as a succession of object stages or phases. The ontology has a focus on conceptual simplicity, so that the number of basic (primitive) concepts and relations is as small as possible in order to simplify the theory.

## PROTON

PROTON (PROTo ONtology) is a basic subsumption hierarchy which provides coverage of most of the upper-level concepts necessary for semantic annotation, indexing, and retrieval.

Chapter 13

# TIME-ITEM & TOVE Project

## TIME-ITEM

**TIME-ITEM** is an ontology of *Topics* that describes the content of undergraduate medical education. TIME is an acronym for "Topics for Indexing Medical Education"; ITEM is an acronym for "Index de thèmes pour l'éducation médicale." Version 1.0 of the taxonomy has been released and the web application that allows users to work with it is still under development. Its developers are seeking more collaborators to expand and validate the taxonomy and to guide future development of the web application.

### *History*

The development of TIME-ITEM began at the University of Ottawa in 2006. It was initially developed to act as a content index for a curriculum map being constructed there. After its initial presentation at the 2006 conference of the Canadian Association for Medical Education, early collaborators included the University of British Columbia, McMaster University and Queen's University.

### *Features*

The TIME-ITEM ontology is unique in that it is designed specifically for undergraduate medical education. As such, it includes fewer strictly biomedical entries than other common medical vocabularies (such as MeSH or SNOMED CT) but more entries relating to the medico-social concepts of communication, collaboration, professionalism, etc.

Topics within TIME-ITEM are arranged poly-hierarchically, meaning any Topic can have more than one parent. Relationships are established based on the logic that learning about a Topic contributes to the learning of all its parent Topics.

In addition to housing the ontology of Topics, the TIME-ITEM web application can house multiple Outcome frameworks. All Outcomes, whether private Outcomes entered by single institutions or publicly available medical education Outcomes (such as CanMeds 2005) are hierarchically linked to one or more Topics in the ontology. In this way, the contribution of each Topic to multiple Outcomes is made explicit.

The structure of the XML documents exported from TIME-ITEM (which contain the hierarchy of Outcomes and TIME-ITEM Topics) is being developed alongside the MedBiquitous Competency standards.

The taxonomy currently exists in English only but translation to Canadian French is in progress.

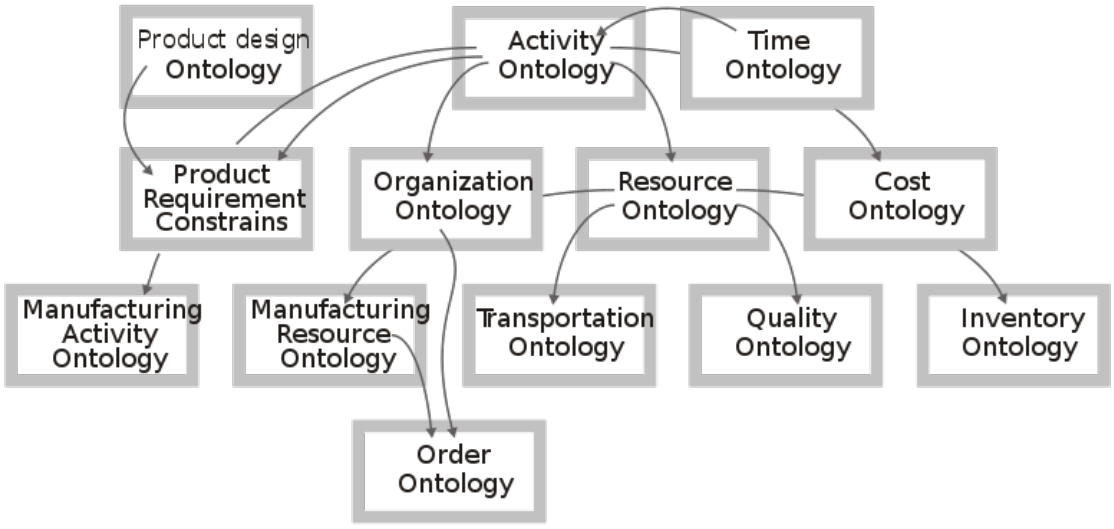
## *Applications*

TIME-ITEM is intended to be a general-use ontology for medical education informatics. It has two primary potential applications:

1. Inclusion in curriculum maps. By mapping learning objects or sessions to TIME-ITEM Topics in a curriculum map, the map becomes searchable for both granular and broad concepts. If one or more Outcome frameworks are included with the Topic ontology, then the contribution of each curricular element to one or more Outcomes is made explicit. This also facilitates curriculum evaluation in terms of one or more outcome frameworks.
2. Indexing learning, assessment and portfolio objects. Metatagging learning objects or assessment objects with a controlled vocabulary enhances the organization and retrieval of objects from a repository. Metatagging electronic portfolio entries allows one to show how multiple entries "add up" to a demonstration of competence with respect to certain educational outcomes.

The possibility of expanding or modifying TIME-ITEM for use in postgraduate/continuing medical education and in nursing education is currently being explored.

# TOVE Project




Toronto Virtual Enterprise Ontologies, Fox and Gruninger (1998).

The **TOVE project**, acronym of *TOronto Virtual Enterprise* project is a project to develop a ontological framework for enterprise integration (EI) based on and suited for enterprise modeling. In the beginning of the 1990s it was initiated by Mark E. Fox and others at the University of Toronto.

The original goal of the project was fourfold:

- Create a shared representation or ontology of the enterprise that each agent in the distributed enterprise can jointly understand and use
- Define the meaning of each description or semantics
- Implement the semantics in a set of axioms that will enable TOVE to automatically deduce the answer to many "common sense" questions about the enterprise, and
- Define a symbology for depicting a concept in a graphical context

Eventually the ungoing project aims to develop a set of integrated ontologies for the modelling of both commercial and public enterprises.

The model according to Ted Williams (2000) is "multi-level, spanning conceptual, generic and applications layers. The generic and applications layers all also stratified and composed of micro theories spanning, for example, activities, time, resources, constraints, etc. At the generic level. Critical to the TOVE effort is enabling the easy instantiation of the model for a particular enterprise TOVE models will be automatically created as a by product of the enterprise design function". At the turn of the millennium TOVE has been to model a computer manufacturer and an aerospace engineering firm.

The TOVE enterprise models is presented by Fox et al. (1995) as second generation knowledge engineering approach. A first generation knowledge engineering approach "is extracting rules from experts, while second generation is ontology engineering: They develop comprehensive ontologies for all the aspects of an organization they find necessary (necessity is decided based on competency requirements to the model, i.e., what are the questions the model will have to answer, either by ordinary look-up or by deduction). The background of TOVE is clearly knowledge engineering and to some degree Computer Integrated Manufacturing".

Chapter 14

# Semantic Spectrum

The **semantic spectrum** (sometimes referred to as the **ontology spectrum** or the **smart data continuum** or **semantic precision**) is a series of increasingly precise or rather semantically expressive definitions for data elements in knowledge representations, especially for machine use.

At the low end of the spectrum is a simple binding of a single word or phrase and its definition. At the high end is a full ontology that specifies relationships between data elements using precise URIs for relationships and properties.

With increased specificity comes increased precision and the ability to use tools to automatically integrate systems but also increased cost to build and maintain a metadata registry.

Some steps in the semantic spectrum include the following:

1. glossary: A simple list of terms and their definitions. A glossary focuses on creating a complete list of the terminology of domain-specific terms and acronyms. It is useful for creating clear and unambiguous definitions for terms and because it can be created with simple word processing tools, few technical tools are necessary.
2. controlled vocabulary: A simple list of terms, definitions and naming conventions. A controlled vocabulary frequently has some type of oversight process associated with adding or removing data element definitions to ensure consistency. Terms are often defined in relationship to each other.
3. data dictionary: Terms, definitions, naming conventions and one or more representations of the data elements in a computer system. Data dictionaries often define data types, validation checks such as enumerated values and the formal definitions of each of the enumerated values.

4. data model: Terms, definitions, naming conventions, representations and one or more representations of the data elements as well as the beginning of specification of the relationships between data elements including abstractions and containers.
5. taxonomy: A complete data model in an inheritance hierarchy where all data elements inherit their behaviors from a single "super data element". The difference between a data model and a formal taxonomy is the arrangement of data elements into a formal tree structure where each element in the tree is a formally defined concept with associated properties.
6. ontology: A complete, machine-readable specification of a conceptualization using URLs for all data elements, properties and relationship types. The W3C standard language for representing ontologies is the Web Ontology Language (OWL). Ontologies frequently contain formal business rules formed in discrete logic statements that relate data elements to each another.

## *Typical questions for determining semantic precision*

The following is a list of questions that may arise in determining semantic precision.

correctness
: How can correct syntax and semantics be enforced? Are tools (such as XML Schema) readily available to validate syntax of data exchanges?

adequacy/expressivity/scope
: Does the system represent everything that is of practical use for the purpose? Is an emphasis being placed on data that is externalized (exposed or transferred between systems)?

efficiency
: How efficiently can the representation be searched / queried, and - possibly - reasoned on?

complexity
: How steep is the learning curve for defining new concepts, querying for them or constraining them? are there appropriate tools for simplifying typical workflows?

translatability
: Can the representation easily be transformed (e.g. by Vocabulary-based transformation) into an equivalent representation so that semantic equivalence is ensured?

## Determining location on the semantic spectrum

Many organizations today are building a metadata registry to store their data definitions and to perform metadata publishing. The question of where they are on the semantic spectrum frequently arises. To determine where your systems are, some of the following questions are frequently useful.

1. Is there a centralized glossary of terms for the subject matter?
2. Does the glossary of terms include precise definitions for each terms?

3. Is there a central repository to store data elements that includes data types information?
4. Is there an approval process associated with the creation and changes to data elements?
5. Are coded data elements fully enumerated? Does each enumeration have a full definition?
6. Is there a process in place to removed duplicate or redundant data elements from the metadata registry?
7. Is there one or more classification schemes used to classify data elements?
8. Are document exchanges and web services created using the data elements?
9. Can the central metadata registry be used as part of a Model-driven architecture?
10. Are their staff members trained to extract data elements are reuse metadata structures?

## *Strategic nature of semantics*

Today, much of the World Wide Web is stored as Hypertext Markup Language. Search engines are severely hampered by their inability to understand the meaning of published web pages. These limitations have led to the advent of the Semantic web movement.

In the past, many organizations that created custom database application used isolated teams of developers that did not formally publish their data definitions. These teams frequently used internal data definitions that were incompatible with other computer systems. This made Enterprise Application Integration and Data warehousing extremely difficult and costly. Many organizations today require that teams consult a centralized data registry before new applications are created.

The job title of an individual that is responsible for coordinating an organization's data is a Data architect.

## *History*

The first reference to this term was at the 1999 AAAI Ontologies Panel. The panel was organized by Chris Welty, who at the prodding of Fritz Lehmann and in collaboration with the panelists (Fritz, Mike Uschold, Mike Gruninger, and Deborah McGuinness) came up with a "spectrum" of kinds of information systems that were, at the time, referred to as ontologies. The "ontology spectrum" picture appeared in print in the introduction to *Formal Ontology and Information Systems: Proceedings of the 2001 Conference.* The ontology spectrum was also featured in a talk at the Semantics for the Web meeting in 2000 at Dagstuhl by Deborah McGuinness. McGuinness produced a paper describing the points on that spectrum that appeared in the book that emerged (much later) from that workshop called "Spinning the Semantic Web." Later, Leo Obrst extended the spectrum into two dimensions (which technically is not really a spectrum anymore) and added a lot more detail, which was included in his book, *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management.*

The concept of the Semantic precision in business systems was popularized by Dave McComb in his book *Semantics in Business Systems: The Savvy Managers Guide* published in 2003 where he frequently uses the term **Semantic Precision**.

This discussion centered around a 10 level partition that included the following levels (listed in the order of increasing semantic precision):

1. Simple Catalog of Data Elements
2. Glossary of Terms and Definitions
3. Thesauri, Narrow Terms, Relationships
4. Informal "Is-a" relationships
5. Formal "Is-a" relationships
6. Formal instances
7. Frames (properties)
8. Value Restrictions
9. Disjointness, Inverse, Part-of
10. General Logical Constraints

Note that there was a special emphasis on the adding of formal *is-a* relationships to the spectrum which seems to have been dropped.

The company Cerebra has also popularized this concept by describing the data formats that exist within an enterprise in their ability to store semantically precise metadata. Their list includes:

1. HTML
2. PDF
3. Word Processing documents
4. Microsoft Excel
5. Relational databases
6. XML
7. XML Schema
8. Taxonomies
9. Ontologies

What the concepts share in common is the ability to store information with increasing precision to facilitate intelligent agents.

**Chapter 15**

# ISO 15926

The **ISO 15926** is titled: "*Industrial automation systems and integration—Integration of life-cycle data for process plants including oil and gas production facilities*" is a standard for data integration, sharing, exchange, and hand-over between computer systems.

This title is regarded too narrow by the present ISO 15926 developers. Having developed a generic data model and Reference Data Library for process plants, it turned out that this subject is already so wide, that actually any state information may be modelled with it.

## *History*

In 1991 a European Union ESPRIT-, named ProcessBase, started. The focus of this research project was to develop a data model for lifecycle information of a facility that would suit the requirements of the process industries. At the time that the project duration had elapsed, a consortium of companies involved in the process industries had been established: EPISTLE (European Process Industries STEP Technical Liaison Executive). Initially individual companies were members, but later this changed into a situation where three national consortia were the only members: PISTEP (UK), POSC/Caesar (Norway), and USPI-NL (Netherlands). (later PISTEP merged into POSC/Caesar, and USPI-NL was renamed to USPI).

EPISTLE took over the work of the ProcessBase project. Initially this work involved a standard called ISO 10303-221 (referred to as "AP221"). In that AP221 we saw, for the first time, an Annex M with a list of standard instances of the AP221 data model, including types of objects. These standard instances would be for reference and would act as a knowledge base with knowledge about the types of objects. In the early nineties EPISTLE started an activity to extend Annex M to become a library of such object classes and their relationships: STEPlib. In the STEPlib activities a group of approx. 100 domain experts from all three member consortia, spread over the various expertises (e.g. Electrical, Piping, Rotating equipment, etc), worked together to define the "core classes".

The development of STEPlib was extended with many additional classes and relationships between classes and published as Open Source data. Furthermore, the concepts and relation types from the AP221 and ISO 15926-2 data models were also added to the STEPlib dictionary. This resulted in the development of Gellish English, whereas STEPlib became the Gellish English dictionary. Gellish English is a structured subset of natural English and is a modeling language suitable for knowledge modeling, product modeling and data exchange. It differs from conventional modeling languages (meta languages) as used in information technology as it not only defines generic concepts, but also includes an English dictionary. The semantic expression capability of Gellish English was significantly increased by extending the number of relation types that can be used to express knowledge and information.

For modelling-technical reasons POSC/Caesar proposed another standard than ISO 10303, called ISO 15926. EPISTLE (and ISO) supported that proposal, and continued the modelling work, thereby writing Part 2 of ISO 15926. This Part 2 has official ISO IS (International Standard) status since 2003.

POSC/Caesar started to put together their own RDL (Reference Data Library). They added many specialized classes, for example for ANSI (American National Standards Institute) pipe and pipe fittings. Meanwhile STEPlib continued its existence, mainly driven by some members of USPI. Since it was clear that it was not in the interest of the industry to have two libraries for, in essence, the same set of classes, the Management Board of EPISTLE decided that the core classes of the two libraries shall be merged into Part 4 of ISO 15926. This merging process has been finished. Part 4 should act as reference data for part 2 of ISO 15926 as well as for ISO 10303-221 and replaced its Annex M. On June 5, 2007 ISO 15926-4 was signed of as a TS (Technical Specification).

In 1999 the work on Part 7 started. Initially this was based on XML Schema (the only useful W3C Recommendation available then), but when OWL became available it was clear that that provided a far more suitable environment for Part 7. Part 7 passed the first ISO ballot by the end of 2005, and an implementation project started. A formal ballot for TS (Technical Specification) was planned for December 2007.

## *The standard*

ISO 15926 has 7 parts:

- Part 1 - Introduction, information concerning engineering, construction and operation of production facilities is created, used and modified by many different organizations throughout a facility's lifetime. The purpose of ISO 15926 is to facilitate integration of data to support the lifecycle activities and processes of production facilities.

- Part 2 - Data Model. a generic 4D model that can support all disciplines, supply chain company types and life cycle stages, regarding information about functional

requirements, physical solutions, types of objects and individual objects as well as activities.

- Parts 4 , 5,6 - Reference Data, the terms used within facilities for the process industry.

- Part 7 - Implementation methods for the integration of distributed systems, defining an implementation architecture that is based on the W3C Recommendations for the Semantic Web.

## Description

The model and the library are suitable for representing lifecycle information about technical installations and their components.

They can also be used for defining the terms used in product catalogs in e-commerce. Another, more limited, use of the standard is as a reference classification for harmonization purposes between shared databases and product catalogues that are not based on ISO 15926.

The purpose of ISO 15926 is to provide a Lingua Franca for computer systems, thereby integrating the information produced by them. Although set up for the process industries with large projects involving many parties, and involving plant operations and maintenance lasting decades, the technology can be used by anyone willing to set up a proper vocabulary of reference data in line with Part 4.

In Part 7 the concept of Templates is introduced. These are semantic constructs, using Part 2 entities, that represent a small piece of information. These constructs then are mapped to more efficient classes of n-ary relations that interlink the Nodes that are involved in the represented information.

These Node and Template instances are stored in Façades. A Façade is an RDF quad store, set up to a standard schema and API, as defined in Part 7. Any Façade only stores the data for which the Façade owner is responsible.

The Façade API has methods for:

- population_import = parsing, validating, and converting (to quads) an incoming RDF/XML file
- population = entering instances of Nodes and Templates in a Façade, removing duplicates, terminating replaced data
- hand-over = physically moving instances of Nodes and Templates from one Façade to another
- replace_uri = optionally adds a suffix to IDs of handed over objects, creates cross-reference oldURI vs newURI

- handover_export = creating a transfer file and invoking 'population_import' at the receiving Façade
- format_message = generating standard message with included documents, invoking 'gendoc' method
- gendoc = generate a rendering of all included documents in PDF format
- send_message = sending a Message containing one or more Documents from one Façade to another
- parse_message = parsing of Message and included Documents, dereferencing of all used URIs, cross-referencing between local IDs and (foreign) templates in which these are used.

Each participating computer system maps its data from its internal format to such ISO-standard Node and Template instances. These are stored in a System Façade, each system its own Façade.

Data can be "handed over" from one Façade to another in cases where data custodianship is handed over (e.g. from a contractor to a plant owner, or from a manufacturer to the owners of the manufactured goods). Hand-over can be for a part of all data, whilst maintaining full referential integrity.

Façades can be set up for the consolidation of data by handing over data produced by various participating computer systems and stored in their System Façades. Examples are: a Façade for a project discipline, a project, a plant, or even for a company in a fiscal year).

Documents are user-definable. They are, in essence, only a structure containing cells that make reference to instances of Templates. This represents a view on all lifecycle data: since the data model is a 4D (space-time) model, it is possible to present the data that was valid at any given point in time, thus providing a true historical record. It is expected that this will be used for Knowledge Mining.

Data can be queried by means of SPARQL. In any implementation a restricted number of Façades can be involved, with different access rights. This is done by means of creating a matrix called a CPF (= Confederation of Participating Façades). An Ontology Browser allows for access to one or more Façades in a given CPF, depending on the access rights.

## *Projects and applications*

There are a number of projects working on the extension of the ISO 15926 standard in different application areas.

### Capital Intensive projects

Within the application of Capital Intensive projects, two cooperating implementation projects are running at the moment (May 2009):

- The ADI Project of FIATECH, to build the tools (which will then be made available in the public domain)
    - The tools and deliverables can been seen on the ISO 15926 knowledge base:

- The IDS Project of POSC Caesar Association, to define product models required for data sheets

- A joint ADI-IDS project is the ISO 15926 WIP

## Upstream Oil and Gas industry

The Norwegian Oil Industry Association (OLF) has decided to use ISO 15926 (also known as the Oil and Gas Ontology) as the instrument for integrating data across disciplines and business domains for the Upstream Oil and Gas industry. It is seen as one of the enablers of what has been called the next (or second) generation of Integrated operations, where a better integration across companies is the goal.

The following projects are currently running (May 2009):

- The Integrated Operations in the High North (IOHN) project is working on extending ISO 15926 to handle real-time data transmission and (pre-)processing to enable the next generation of Integrated Operations.
- The Environment Web project to include environmental reporting terms and definitions as used in EPIM's EnvironmentWeb in ISO 15926.

Finalised projects include:

- The Integrated Information Platform (IIP) project working on establishing a real-time information pipeline based on open standards. It worked among others on:
    - Daily Drilling Report (DDR) to including all terms and definitions in ISO 15926. This standard became mandatory on February 1, 2008 for reporting on the Norwegian Continental Shelf by the Norwegian Petroleum Directorate (NPD and Safety Authority Norway (PSA). NPD says that the quality of the reports has improved considerably since.
    - Daily Production Report (DPR) to including all terms and definitions in ISO 15926. This standard was tested successfully on the Valhall (BP-operated) and Åsgard (StatoilHydro-operated) fields offshore Norway. The terminology and XML schemata developed have also been included in Energistics' PRODML standard.

## *Some technical background*

One of the main requirements was (and still is) that the scope of the data model covers the entire lifecycle of a facility (e.g. oil refinery) and its components (e.g. pipes, pumps and their parts, etc.). Since such a facility over such a long time entails many different

types of activities on a myriad of different objects it became clear that a generic and data-driven data model would be required.

A simple example will illustrate this. There are thousands of different types of physical objects in a facility (pumps, compressors, pipes, instruments, fluids, etc). Each of these has many properties. If all combinations would be modelled in a "hard-coded" fashion, the number of combinations would be staggering, and unmanageable.

The solution is a "template" that represents the semantics of: "This object has a property of X yyyy" (where yyyy is the unit of measure). Any instance of that template refers to the applicable reference data:

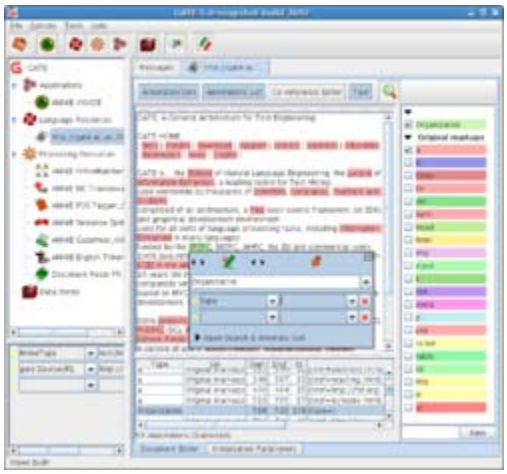
- physical object (e.g. my Induction Motor)
- indirect property type (e.g. the class "cold locked rotor time")
- base property type (here: time)
- scale (here: seconds)

Without being able to make reference to those classes, via the Internet, it will be impossible to express this information.

Chapter 16

# General Architecture for Text Engineering

GATE



GATE Developer v5 main window

| | |
|---|---|
| **Developer(s)** | GATE research team, Dept. Computer Science, University of Sheffield |
| **Initial release** | 1995; 16 years ago |
| **Stable release** | 6.0 (November 8, 2010; 4 months ago) [+/−] |
| **Preview release** | 6.1 (2011-04-02 (Nightly builds released everyday)) [+/−] |
| **Written in** | Java |
| **Operating system** | Cross-platform |
| **Available in** | English |
| **Type** | Text mining Information |

| | |
|---|---|
| | Extraction |
| **License** | LGPL |

**General Architecture for Text Engineering** or **GATE** is a Java suite of tools originally developed at the University of Sheffield beginning in 1995 and now used worldwide by a wide community of scientists, companies, teachers and students for all sorts of natural language processing tasks, including information extraction in many languages.

GATE includes:

- an IDE, GATE Developer: an integrated development environment for natural language processing components bundled with a very widely used information extraction system and a comprehensive set of other plugins
- a web app, GATE Teamware: a collaborative annotation environment for factory-style semantic annotation projects built around a workflow engine and a heavily-optimised backend service infrastructure
- a framework, GATE Embedded: an object library optimised for inclusion in diverse applications giving access to all the services used by GATE Developer and more
- an architecture: a high-level organisational picture of language processing software composition
- a process for the creation of robust and maintainable services.

Under development:

- a cloud computing solution for hosted large-scale text processing, GATE Cloud

GATE aims to remove the necessity for solving common engineering problems before doing useful research, or re-engineering before deploying research results into applications. Core functions of GATE take care of the lion's share of the engineering:

- modelling and persistence of specialised data structures
- measurement, evaluation, benchmarking
- visualisation and editing of annotations, ontologies, parse trees, etc.
- a finite state transduction language for rapid prototyping and efficient implementation of shallow analysis methods (JAPE, see below)
- extraction of training instances for machine learning
- pluggable machine learning implementations (Weka, SVM Light, an in-house uneven margins SVM implementation and more.)

On top of the core functions, GATE includes components for diverse natural language processing tasks, e.g. parsers, morphology, tagging, information retrieval tools, information extraction components for various languages, and many others. It has been widely applied in fields such as bioinformatics and others. GATE Developer and Embedded are supplied with an information extraction system (ANNIE) which has been

adapted and evaluated very widely (numerous industrial systems, research systems evaluated in MUC, TREC, ACE, DUC, Pascal, NTCIR, etc.). ANNIE is often used to create RDF or OWL (metadata) for unstructured content (semantic annotation). GATE has been compared to NLTK, R and RapidMiner. As well as being widely used in its own right, it forms the basis of the KIM semantic platform.

GATE community and research has been involved in several European research projects including TAO, SEKT, NeOn, Media-Campaign, Musing, Service-Finder, LIRICS and KnowledgeWeb, as well as many other projects.

As of December 4, 2009, 691 people are on the gate-users mailing list at SourceForge.net, and 98,858 downloads from SourceForge are recorded since the project moved to SourceForge in 2005. The paper "GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications" has received over 800 citations in the seven years since publication (according to Google Scholar). Books covering the use of GATE, in addition to the GATE User Guide, include "Building Search Applications: Lucene, LingPipe, and Gate", by Manu Konchady, and "Introduction to Linguistic Annotation and Text Analytics", by Graham Wilcock.

## *Features*

GATE includes an information extraction system called ANNIE (A Nearly-New Information Extraction System) which is a set of modules comprising a tokenizer, a gazetteer, a sentence splitter, a part of speech tagger, a named entities transducer and a coreference tagger. ANNIE can be used as-is to provide basic information extraction functionality, or provide a starting point for more specific tasks.

Languages currently handled in GATE include English, Spanish, Chinese, Arabic, French, German, Hindi, Italian, Cebuano, Romanian, Russian.

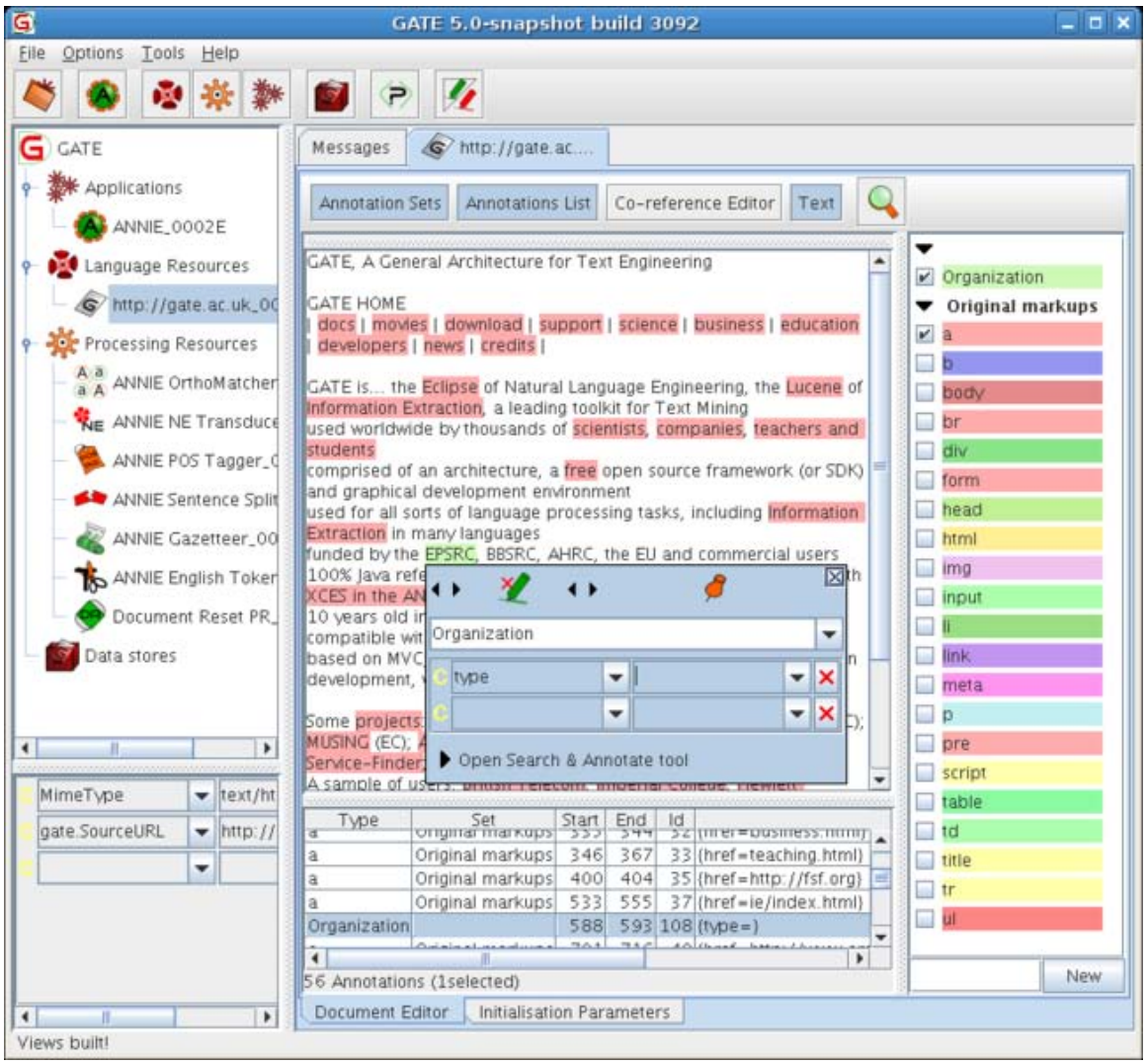
Plugins are included for machine learning with Weka, RASP, MAXENT, SVM Light, as well as a fast LibSVM integration and an in-house perceptron implementation, for managing Ontologies like WordNet, for querying search engines like Google or Yahoo, for part of speech tagging with Brill or TreeTagger, and many more.

GATE can handle input in various formats, such as TXT, HTML, XML, Doc, PDF documents, and Java Serial, PostgreSQL, Lucene, Oracle Databases with help of RDBMS storage over JDBC.

It also uses the JAPE (Java Annotation Patterns Engine) language for building rules in order to annotate documents with tags. JAPE stands for "Java Annotation Patterns Engine". JAPE provides finite state transduction over annotations based on regular expressions. JAPE is a version of CPSL – Common Pattern Specification Language. JAPE transducers are used within GATE to manipulate annotations on text. Documentation is provided in the GATE User Guide. A tutorial has also been written by Press Association Images.

## *GATE Developer*

GATE Developer is the GATE graphical user interface. It is analogous to systems like Mathematica for mathematicians, or Eclipse for Java programmers, providing a convenient graphical environment for research and development of language processing software. As well as being a powerful research tool in its own right, it is also useful in conjunction with GATE Embedded (the GATE API by which GATE functionality can be included in applications); for example, GATE Developer can be used to create applications that can then be embedded via the API.




GATE 5 main window.

The GATE Developer GUI consists of a top menu and row of icons, a left vertical resources tree, a central-right tabbed pane of the resource viewers and a message field at the bottom.

The resources tree and the menu are use to load, save and run resources. The resources tree display the loaded resources and allows to show a resource in a resource viewer by double-clicking on it or pressing Enter key.

Each loaded resource can be displayed in a specific resource viewer that take most of the space in the GUI.

Here you can see the document viewer use to display a document and its annotations. In pink are <A> hyperlink annotations from an HTML file. The right list is the annotation sets list and the bottom table is the annotation list. In the center is the annotation editor window.

## *GATE Teamware*

Teamware is a web-based management platform for collaborative annotation & curation. GATE Teamware delivers a multi-function user interface over the Internet for viewing, adding and editing text annotations. The web-based management interface allows for project set-up, tracking, and management:

- Loading document collections (a "corpus" or "corpora")
- Creating re-usable project templates
- Initiating projects based on templates
- Assigning project roles to specific users
- Monitoring progress and various project statistics in real time
- Reporting of project status, annotator activity and statistics
- Applying GATE-based processing routines (automatic annotations or post-annotation processing)