A Framework for Understanding and Classifying Ontology Applications

Mike Uschold
The Boeing Company
P.O. Box 3707,m/s 7L-40
Seattle, WA USA
98124-2207
michael.f.uschold@boeing.com

Robert Jasper
The Boeing Company
P.O. Box 3707,m/s 7L-40
Seattle, WA USA
98124-2207
robert.j.jasper@boeing.com

Abstract

In[1] this paper, we draw attention to common goals and supporting technologies of several relatively distinct communities to facilitate closer cooperation and faster progress. The common thread is the need for sharing the meaning of terms in a given domain, which is a central role of ontologies. The different communities include ontology research groups, software developers and standards organizations. Using a broad definition of 'ontology', we show that much of the work being done by those communities may be viewed as practical applications of ontologies.

To achieve this, we present a framework for understanding and classifying ontology applications. We identify three main categories of ontology applications: 1) neutral authoring, 2) common access to information, and 3) indexing for search. In each category, we identify specific ontology application scenarios. For each, we indicate their intended purpose, the role of the ontology, the supporting technologies and who the principal actors are and what they do. We illuminate the similarities and differences between scenarios.

Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5) Stockholm, Sweden, August 2, 1999

(V.R. Benjamins, B. Chandrasekaran, A. Gomez-Perez, N. Guarino, M. Uschold, eds.)

http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-18/

[1] The order of authors was determined by a coin flip.

1 Introduction

Until recently, there had been a dearth of ontology applications reported by the AI ontology community. This has begun to change; in the past year or two, there have been a flurry of papers reporting attempts and some successes at applying ontologies — especially in the area of search and retrieval of information repositories, for example, [8]. And yet, outside the AI ontology community, industry has been regularly using ontologies successfully (even though they may not call them ontologies).

There is a common thread that binds these different communities: the need to overcome barriers created by disparate vocabularies, approaches, representations, and tools in their respective contexts. There is a need to share meaning of terms in a given domain. Achieving a shared understanding is accomplished by agreeing on an appropriate way to conceptualize the domain, and then to make it explicit in some language. The result, an ontology, can be applied in a wide variety of contexts for various purposes.

These groups strive to overcome the barriers noted in the previous paragraph; ironically, the same underlying issues also create barriers to closer cooperation between ontology research groups, software developers and standards organizations who are addressing similar problems. This paper aims to lower these barriers by identifying and highlighting the commonality among these communities, and pointing out important differences. We do this by providing a framework for understanding and classifying ontology applications. In creating this framework, we propose a common nomenclature, that we hope will enable workers in the different communities to overcome terminological confusion. We do not try to reconcile the differences between the communities; we instead highlight the commonality between these groups through the use of a single framework.

Another important goal of this work is to lay the foun-

dation for identifying and expressing guidelines for how to use ontologies to achieve specific benefits.

1.1 Terminology & Acronyms

For the purposes of this paper, we take a 'lowest common denominator' view of the notion of an ontology. We do not aim to define it, instead we adopt the following characterization, quoted from [15] (our emphasis):

> "An ontology may take a variety of forms, but necessarily it will include a *vocabulary of terms*, and some *specification of their meaning*. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the domain and constrain the possible interpretations of terms."

As noted below, the degree to which meaning is specified varies greatly. This broad interpretation helps to show how both the goals and the technologies developed to achieve them are similar across the different communities. For example, common goals include reuse and interoperability. Common technologies include special purpose modeling languages (e.g., Ontolingua [2], EXPRESS [11, 10] and IDL) and translation tools. Thus, we can easily view a number of standardization efforts (e.g., STEP, OMG[2]) as practical applications of ontologies.

Application: refers to the application that makes use of or benefits from the ontology (possibly indirectly).

OMG**:** Object Management Group

CORBA**:** Common Object Request Broker Architecture

STEP**:** STandard for the Exchange of Product model data; an informal name for the ISO-10303 family of standards

EXPRESS**:** an object-flavored language for specifying information models, originally developed as part of the STEP standard.

2 Overview of the Framework

The main part of the framework consists of a set of *ontology application scenarios*. By this we mean, a particular situation or context in which an ontology is applied in a particular way to achieve one or more specific benefits. We identify three main categories of ontology applications: 1) neutral authoring, 2) common access to information, and 3) indexing for search. For each category, we identify one or more specific application scenarios. For example, in the neutral authoring category, there are two scenarios, one for authoring an ontology, and the other for authoring operational data.

[2] See www.omg.org

Each scenario is illustrated with a simple diagram. Many of the scenarios have important variations, that we also call attention to. The scenarios and variations are illustrated with examples from the diverse communities.

To achieve our goal of enabling scenarios to be easily compared, we present each in a uniform way using consistent terminology. Each scenario is characterized by the following, which give rise to the key dimensions of our framework:

1. intended purpose or benefits
2. the role of the ontology
3. the actors required to implement the scenario
4. supporting technologies
5. the maturity level

Two additional distinctions that play an important role in some scenarios, are:

- how meaning of terms is represented (e.g., informal or formal)
- sharing vs exchange (e.g., pass by reference or pass by value)

In the remainder of this section, we describe the key dimensions and dinstinctions which form the basis for our framework. In § 3 we present the ontology application scenarios.

2.1 Framework Dimensions

2.1.1 Purpose and Benefits

Fundamentally, ontologies are used to improve communication between either humans or computers. Broadly, these may be grouped into the following three areas: to assist in communication between human agents, to achieve interoperability, or to improve the process and/or quality of engineering software systems. The following is adapted from [15].

Communication between *people*. Here, an unambiguous but informal ontology may be sufficient.

Inter-Operability among *computer systems* achieved by translating between different modeling methods, paradigms, languages and software tools; here, the ontology is used as an interchange format.

Systems Engineering Benefits: In particular,

Re-Usability: the ontology is the basis for a formal encoding of the important entities, attributes, processes and their inter-relationships in the domain of interest. This formal representation may be (or become so by automatic translation) a re-usable and/or shared component in a software system.

Search: an ontology may be used as meta-data serving as an index into a repository of information.

Reliability: A formal representation also makes possible the automation of consistency checking resulting in more reliable software.

Specification: the ontology can assist the process of identifying requirements and defining a specification for an IT system (knowledge based, or otherwise).

Maintenance: use of ontologies in system development, or as part of an end application, can render maintenance easier in a number of ways. Systems which are built using explicit ontologies serve to improve documentation of the software which reduces maintenance costs. Maintenance is also an important benefit if an ontology is used as a neutral authoring language with multiple target languages - it only has to be maintained in one place.

Knowledge Acquisition: speed and reliability may be increased by using an existing ontology as the starting point and basis for guiding knowledge acquisition when building knowledge-based systems.

2.1.2 Role of the Ontology

Ontology application scenarios vary in how the ontology itself is used. This is further complicated by the fact that in a given scenario, it is frequently possible to think of more than one ontology being involved. For example, when Ontolingua is used, (1) the frame ontology is used as a basis for expressing (2) the domain ontology. An ontology application scenario needs to be clear about which ontology is being used and how. To facilitate this, we introduce three roles that information can play in one of our scenarios. These can also be thought of as information levels.

L_0**:** *Operational Data* a role that information plays whereby the information is consumed and produced by applications during runtime. Information at L_0 is written using terms from a vocabulary defined at L_1.

L_1**:** *Ontology* a role that information plays, whereby the information specifies terms and definitions for important concepts in some domain. An ontology typically is used in the development processes to create applications.[3] Information at L_1 provides a vocabulary for the language used to author information at L_0.

L_2**:** *Ontology Representation Language* a role that information plays whereby the information is used by ontology authors or application developers, during the development process, to write ontologies at level L_1. The information at L_2 is used to author information at L_1.

Importantly, the same information can play different roles at different times depending on the context. For example, a schema in EXPRESS plays the role of an ontology from the viewpoint of an end-user application. From the viewpoint of an application development tool (e.g., an EXPRESS compiler), the same information plays the role of operational data.

To avoid this kind of potential confusion in this paper, we focus on end-user applications where information plays the role of operational data (L_0). With this assumption, the following are examples of information typical at each level:

L_0 operational data (e.g., a particular part, a process description)

L_1 an ontology (e.g., AP203, PIF)

L_2 an ontology representation language (e.g., EXPRESS, Ontolingua)

To share or exchange information at L_n requires reference to a model at L_{n+1}. For example, sharing Ontolingua ontologies (at L_1) requires development tools that can parse the syntax of Ontolingua (at L_2). Another good example arises in the context of the STEP family of standards. STEP defines a standard for exchange of schema instances via clear text encoding (ISO-10303-21, 1994) which is at level L_0. Exchange at this level requires a schema at L_1 written in EXPRESS [11, 10] which is at level L_1. Similar analogies exist for object sharing and exchange standards (e.g., OMG).

Information at the same level can represented using more than one syntax (e.g., an ontology in Ontolingua versus Loom). It is also possible that you can use the same syntax to represent information at different levels. For example, a class definition at L_1 and an instance definition at L_0 may both be expressed in the syntax of Ontolingua. This is because some primitives of Ontolingua (e.g., define-instance) can be used as part of an L_1 language.

In addition to the syntax, terms may require mapping within or between levels. For example the term `define-class` in Ontolingua may be mapped to the term `defconcept` in Loom.

2.1.3 Actors

Each scenario involves a set of actors. Each actor represents a role that a person or application may play. A person or application may play more than one role in a scenario. Actors may play either a primary or a secondary role in a scenario. The follow list describes the actors:

Ontology Author: (OA) the role of the author of an ontology. This role is usually played by a person.

[3] Some applications may actually "discover" information at this level during operation of the application. This requires some intelligence on the part of the application to "learn" the ontology prior to actual interpretation of the information at L_1.

(Operational) Data Author: (DA) the role of the author of operational data in the language which uses and/or is defined in terms of the vocabulary of the ontology.

Application Developer: (AD) the role of the developer of the Application.

Application User: (AU) the role of the user of the Application.

Knowledge Worker: (KW) the role of the person who makes use of the knowledge.

2.1.4 Supporting Technologies

A great variety of technologies exist that can support ontology applications. These include, but are not limited to:

- Ontology representation languages-(e.g., UML, Express, Ontolingua, XML)
- Knowledge interchange languages: (e.g., KIF, PIF[7], CDIF)
- Translation tools: (e.g., Ontolingua translators, CDIF-tools, StepTools, ... (lots!))
- Distributed Objects: (e.g., CORBA, COM)

2.1.5 Maturity Level

We indicate the degree to which applications and the supporting technologies using a given scenario are mature. At one extreme a scenario may be an untested idea, a specification for a class of potential applications. Systems that are already implemented very from tiny scale demonstrations of feasibility in a research environment to fielded applications in a commercial environment.

2.2 Other Important Distinctions

2.2.1 Representation of Meaning

How meaning in an ontology is represented varies greatly, and turns out to be an important factor in the success of applying ontologies. The simplest ontologies, in this regard, consist of a simple taxonomy of terms. The only meaning is supplied by a single relation which defines the taxonomy. The relation is usually the specialization relationship, but often it is a conglomeration of various relationships such as part-of, or similar-subject-matter. Close inspection of the implicit taxonoomy of Yahoo! reveals that there is no consistent specific meaning of the relationship. At the other extreme, are rigorously formal and carefully axiomatized ontologies such as TOVE [4] and PIF [7].

The meaning captured in an ontology varies both in the *amount* being represented and the degree of *formality* of the representation. The amount of meaning (an attribute of the ontology itself) is directly related to restricting the possible interpretations which serves the primary purpose of reducing ambiguity. The greater the amount of meaning, the more fewer the possible interpretations and the less the ambiguity. Formality (an attributed of the ontology representation language) can vary from natural language to formal logic. We identify four notional points along a continuum of formality:

highly informal: expressed loosely in natural language *e.g.*, many glossaries fit into this category;

structured-informal: expressed in a restricted and structured form of natural language, greatly increasing clarity by reducing ambiguity,
e.g., the text version of the 'Enterprise Ontology' [14] and the glossary of workflow terms produced by the Workflow Management Coalition [9];

semi-formal: expressed in an artificial formally defined language; *e.g.*, the Ontolingua version of the Enterprise Ontology[4];

rigorously formal: meticulously defined terms with formal semantics, theorems and proofs of such properties as soundness and completeness.
e.g., TOVE [4].

Using a formal language may reduce ambiguity, but only if there are sufficient axioms – otherwise it may be as or more ambiguous than a detailed carefully crafted set of definitions in natural language.

For human communication purposes, informal specification of meaning may be preferred. Low ambiguity is also important for humans using ontologies to aid in the development of systems. So formal definitions may be helpful along with informal ones as accompanying documentation. If the ontology is intended to be automatically processed, then ontologies rich in meaning (hi-fat ontologies) present a more challenging task but promise greater rewards. In the short term, lower-fat ontologies (less rich in meaning) are easier to apply in working systems. An excellent example of this is the multiplicity of uses of ontologies as an index into an information repository, both in industry (e.g., Yahoo!) and research. This contrasts with the very challenging task taken on by the PIF project, which is much further from maturing into commercial tools.

2.2.2 Sharing vs Exchange

Depending on the purpose of the ontology, and the specific needs of the application, different architectures will be appropriate for accessing information resources. We distinguish between exchange and sharing using examples from the STEP collection of standards (ISO-10303, 1994). Similar distinctions can be made in other environments.

[4] Available from "http://www.aiai.ed.ac.uk/ enterprise/enterprise/ontology.html"

Sharing: multiple agents (computer or human) reference a common piece of information. The information typically resides outside any of the applications sharing the information. Less common is sharing of a single application's internal data via references that external applications can use. e.g., STEP's Standard Data Access Interface (SDAI) (ISO-10303-22, 1997)

Exchange: multiple applications exchange by passing the data by value (i.e., copying the data) between them. E.g., STEP's clear text encoding standard (ISO-10303-21, 1994).

3 Ontology Application Scenarios

This section describes scenarios for applying ontologies to achieve one or more purposes. These scenarios are abstractions of specific applications of ontologies taken from industry or research. These scenarios are analogous to Ivor Jacobson's use cases [5]. Each scenario includes an overview, which identifies the intended purpose of the ontology, the role of the ontology, who the important actors are, and the supporting technologies. Each is illustrated with a diagram, and includes a concrete example, which typifies technologies or standards that people might use in the scenario. Where appropriate, we identify a number of alternate variations of the main scenario. Finally, we assess the maturity level of each scenario.

We categorize the scenarios into the following three areas:

Neutral Authoring: an information artifact is authored in a single language, and is converted into a different form for use in multiple target systems. Benefits of this approach include knowledge reuse, improved maintainability and long term knowledge retention.

Common Access to Information: information is required by one or more persons or computer applications, but is expressed using unfamiliar vocabulary, or in an inaccessible format. The ontology helps render the information intelligible by providing a shared understanding of the terms, or by mapping between sets of terms. Benefits of this approach include inter-operability, and more effective use and reuse of knowledge resources.

Indexing: an ontology is used as a mechanism for indexing information artifacts. The chief benefit of this approach is faster access to important information resources, which leads to more effective use and reuse of knowledge resources.

4 Scenarios: Neutral Authoring

The basic idea of these scenarios is to author an artifact in a single language, and to have that artifact converted into a different form to be used in multiple target systems. The benefits of this approach include decreased cost of reuse and portability of knowledge across applications, improved application maintainability and long term knowledge retention (e.g., via reduced disruption from changes in vendor formats).

The scenarios in this category differ from each other in a number of ways. First, the authored artifact may be an ontology, or operational data. Second, the process of converting the artifact to a different form varies. It may be direct language to language translation, manual or automated, in which case the translation process may exploit both the syntax and/or semantics of represented concepts. Alternatively, the conversion may best be viewed as design, whereby the ontology is used by the developer as a requirements specification for the target applications. This does not result in a different explicit representation of the ontology, but rather the ontology is implicit in the application. In this case, there is no direct language to language translation. An example of this is the use of the ontologies in the KADS methodology for developing knowledge based systems. Another example is software developed by Specware[17, 18].

4.1 Authoring Ontologies

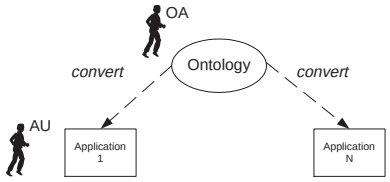


Figure 1: Authoring Ontologies

4.1.1 Overview

The motivation behind authoring neutral ontologies is decreased cost of reuse and increased maintainability of knowledge. To accomplish this, the actors develop an ontology, which they can convert into multiple operational target systems. Supporting technologies include unidirectional ontology translators and software development processes (e.g., KADS). The principle actors are the ontology author and application user.

In this scenario, the ontology author creates an ontology, which they convert into an operational target system (e.g., a knowledge base). Application users then interact with an operational system to perform their desired tasks.

4.1.2 Examples

1. An ontology author creates an ontology (e.g., for titanium alloys) in an ontology authoring language (e.g., Ontolingua). An application developer translates the ontology

into Loom syntax (possibly assisted by automatic translation tools). An application developer directly imports this translated ontology into Loom and it becomes part of the end application, which may contain additional information in its knowledge base. An application user interacts with the final system to answer questions about titanium alloys. This ontology can be reused if another application developer wishes to make use of it in another language, e.g., Prolog. In that case, they will have to translate the ontology into Prolog and proceed as per the Loom example. Note that in this authoring scenario, only one-way translation is required. This contrasts with the case described in the next section, where two way translation is required when an ontology is used as an interchange format.

This example illustrates how to achieve knowledge reuse by virtue of the fact that an ontology authored in a single language can be used in multiple applications.

2. An ontology author creates an ontology using the conceptual modeling language (CML) from the KADS methodology. The application developer uses this ontology as part of the requirements specification when developing the target KBS (e.g., for diagnosing faults). An application user then interacts with the KBS to solve their tasks.

In this example, maintainability is improved because there is an explicit representation of the ontology that the software is based on. Reuse is achieved if the ontology is used for different applications in the same domain.

3. Automated software synthesis into multiple target languages (e.g., using Specware [17, 18]) is a generalization of the neutral authoring language scenario. Application developers play a key role in both development of the ontology and problem statement specification. Typically, the developers semi-automatically refine the specification and ontology into an operational target application.

4.1.3 Variations

A variation on example 2 above, is when the original intent is to build only one application.

4.1.4 Maturity

Totally automated translation of ontologies into operational targets has been difficult and typically relies on translation of idiomatic expressions [2]. For case studies and analysis of some of these problems see [1, 13, 16]. This requires that the ontology author apply the idioms for automatic translation. Semi-automated software synthesis shows some promise, but it has not been a primary focus of the ontology community.

4.2 Authoring Operational Data

Neutral authoring of operational data is similar in structure to neutral ontology authoring. The focus is on authoring and translating operational data rather than an ontology.

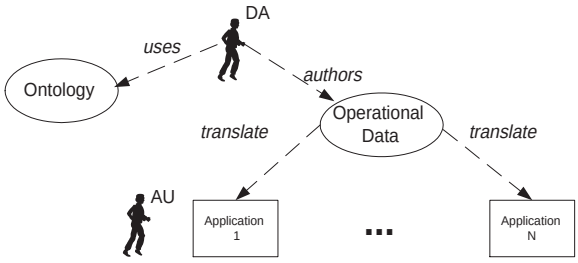


Figure 2: Authoring Operational Data

The main differences are the role the ontology plays in the scenario, and who the primary actors are.

4.2.1 Overview

The motivation behind authoring neutral operational data is improved maintainability and transportability of operational data. To accomplish this, an ontology author (secondary actor) develops an ontology which defines the neutral format used by the primary actor to author the operational data. Tools can then translate this into operational data used by an application. Supporting technologies include unidirectional translators.

In this scenario, a data author creates operational data based on a pre-existing ontology, tools translate these operational data into an operational target system using a unidirectional translator. Application users then interact with the system to perform analysis or query the operational data.

The ontology is originally constructed from careful analysis of the domain of the intended class of target systems, e.g., by identifying and integrating the implicit ontologies for the applications.

4.2.2 Examples

An operational data author uses an ontology (e.g., for workflow systems) to describe a workflow model. Tools translate the description into operation data of various target systems. Application users perform analysis (e.g., critical path) using the translated operational data.

As another example, the Frame Ontology plays the role of ontology for a class of object-oriented representation systems (Loom, Classic, etc.). The engineering math ontology [3] is a set of sentences written using that ontology. Once converted to the appropriate format, this set of sentences plays the role of operational data for the target applications (e.g., Loom). Note that in this example, we are viewing Loom as a system development tool, not an end user application.

This example illustrates the importance of distinguishing the different roles of the information used in these scenarios, and how the same information artifact may play more than one role. It enables us to show the commonality of apparently very different scenarios.

4.2.3 Variations

It may be that only one application and one translator will be used at a time. This arises if the motivation is to reduce risk from changing vendor offerings. If a company maintains their models (i.e. operational data) in a single representation, then when a new vendor format is introduced, it may be easier and more reliable to develop a new translator to convert the neutrally authored artifact to the new format (which might be thousands of lines of code), than to convert the artifact itself directly, (which may be hundreds of thousands of lines). An example of a commercial application using this approach is described in [12].

In the case where point-to-point translators are built, instead of going through an interchange format, the ontology may play an important role in specifying the translator that is created manually. If the ontologies are formal with rich axiomitizations, then there scope for partially automating the construction of the translators and/or in maintaining them when there are changes in eithre language. This is analogous to the use of ontologies in the KADS methodology, where it plays the role of specifying requirements for software.

4.2.4 Maturity

Same remarks apply here as for previous section. Common practice in industry is to build point-to-point translators when the need arises. This may turn out to be more cost effective, depending on the environment.

4.2.5 Closing Remarks

Insofar as a single ontology may be converted and used in many different applications, this is one important way to achieve knowledge reuse. If various systems are based on the same ontology, then this facilitates inter-operation between the systems, should that be required. However, it does not involve sharing or exchange of knowledge between systems. This brings us to the next category.

5 Scenarios: Common Access to Information

The basic idea of this approach is to use ontologies to enable multiple target applications (or humans) to have access to heterogeneous sources of information which is otherwise unintelligible. Benefits of this approach include interoperability, and knowledge reuse.

The scenarios in this category differ in a number of ways. First, the direct consumers of the information may be humans or computer applications. Second, the information artifact may play the role of an ontology, or operational data; the latter may be non-computational (e.g., product data) or computational (e.g., services). Another important distinction is whether the target applications agree on the same shared ontology or whether each has its own local ontology. In the former case, the information is made intelligible via translators, and in the latter case, via ontology mapping rules. Finally, access to the information may be via sharing or exchange.

5.1 Human Communication

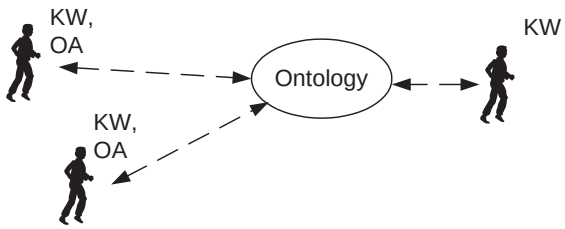


Figure 3: Human Communication

5.1.1 Overview

A major benefit of ontology development is to promote common understanding among knowledge workers. To accomplish this, the authors develop a common shared ontology, which other knowledge workers reference in their work. Non-computational skills such as library classification are valuable in building such ontologies, which commonly take the form of glossaries. Supporting technologies include ontology editors and browsers. The principle actors are the ontology authors and knowledge workers. The information being shared is an ontology.

In this scenario, the ontology authors create an ontology which knowledge workers reference in their work.

5.1.2 Examples

A glossary of terms to enable different working groups, who may have different jargon, to understand each other– (e.g., the workflow management coalition reference document[9]). Producing glossaries, and providing common access for humans to important knowledge assets is a key focus of the Knowledge Management community. Finally, although not in the form of an explicit glossary, the framework in this paper embodies an informal ontology for ontology applications which serves the purpose of enhancing communication between humans who use different terminology.

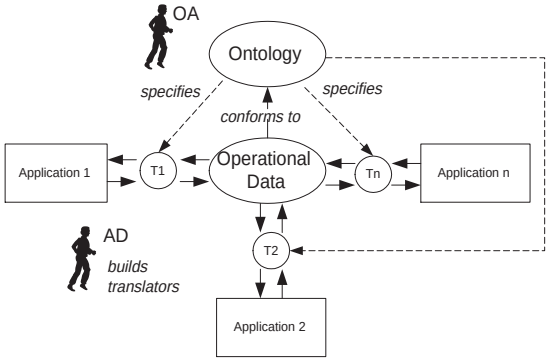
5.1.3 Variations

It may be that the ontology is not the main item of interest, but it enables knowledge workers to better understand documents written using unfamiliar terminology. For example, this paper can also be used to make it easier to understand papers from the different communities being discussed.

5.1.4 Maturity

Informal methods exist for creating informal ontologies. Library classification skills, which have a long history are very appropriate. There may be various tools which offer automated assistence in creating these ontologies, however, we are not aware of them.

5.2 Data Access via Shared Ontology



This scenario indicates how an ontology can be used as an interchange format, to enable common access to operational data.

Figure 4: Data Access via Shared Ontology

5.2.1 Overview

The motivation behind data access via a shared ontology is reducing the cost of multiple applications having common access to data. This may in turn, facilitate inter-operability. This is accomplished through developers agreeing on a shared ontology, which defines a common language for exchanging or sharing operational data. Supporting technologies include translators, parser generators and printers. The principle actors are ontology authors and application developers.

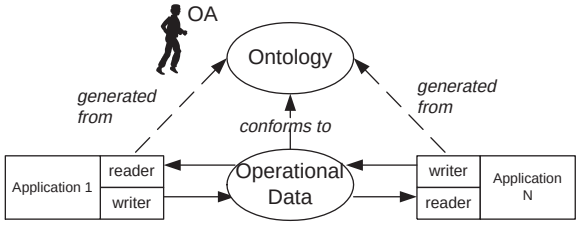
In this scenario, an ontology author creates an ontology, which different application developers agree to use. This defines an interchange format for which two-way translation is required between it and the application formats. Each pair of translators, for a given application, in effect, defines an application interface that can be used to read and write data. Often the translators are manually created, in other cases, explicit readers and writers can be automatically created using parser generators and printers (see variation below). Inter-operation between the multiple applications is made possible by allowing them to access the same information.

5.2.2 Examples

A team of ontology authors created the Process Interchange Format (PIF). The idea is to make a library of process models that are expressed in various application-specific formats available to each of the applications. Currently, they are working on two formats, (IDEF3 and ILOG). This is ongoing research.

EcoCyc [6]is a commercial product that uses a shared ontology to make possible access to various heterogeneous databases in the field of molecular biology. The ontology is a conceptual schema that is an integration of the conceptual schemas for each of the separate databases.

5.2.3 Variations



In this variation translation between formats is achieved by readers and writers which reside in the applications and may be automatically generated

Figure 5: Data Access via Shared Ontology: variation

Figure 4 depicts the natural way to view the situation when there is an explicit linear format that the application uses for saving and loading operational data. The translators are logically separate from the applications and can operate independently. A variation of this is the case where there is *no* such format; instead the internal data structures of the application are used directly by readers and writers internal to the application. So there is no explicit language to language translation per se, but the readers and writers provide the analogue of two-way translation to/from the neutral format (see figure 5).

For example, an ontology author creates a shared ontology for (e.g., for geometric data) in an ontology representation language (e.g., EXPRESS). Application developers use parser and printer generators to generate code in the language du jour (e.g., using the commercial product, StepTools). This provides applications with an API that can be used to read and write data that applications exchange. However, there are no guarantees that the data conforms to all the axioms in the Express schema. Maintaining such consistency is left to the application developers and users.

Another variation data access via a shared ontology is exemplified by the EcoCyc example above. Instead of many applications using their own formats, and translating from one to the other using the ontology as an interchange

format, there is just one application (i.e. database) which uses a single format as specified by the ontology. In this case, there is a one-off translation of the operational data (in this case databases) from the pre-existing formats to the new format. In both this example and the PIF example, there is a very similar process in creating the ontology in the first place. The [possibly implicit] ontologies of several languages used to express information in the same domain are combined into a single neutral format.

There are still other variations. Typically, applications make use of the ontology during the development process by incorporating code generated from the ontology in the application. One variation is to have the application make use of the ontology at runtime (sometimes known as late binding) rather than development time (i.e., early binding).

Another variation involves applications interchanging data via a shared data store. An example is STEP's SDAI interface. A related variation is to only have a single application that reads and writes to data store for purpose of persistence and ease of maintenance.

5.2.4 Maturity

In some contexts, (e.g., product data using EXPRESS) approaches to data access via a shared ontology are relatively mature. Commercial success exists where application developers can agree on shared ontologies. Achieving agreement across a wide variety of applications or industries has been difficult. However, in other contexts, (e.g., PIF) the technology is a long way from being mature.

A number of factors may influence the apparent gulf in maturity between the STEP community and the explicit language to language translation approach (e.g., PIF). Some of the apparent success may be due to shear differences in the amount of effort applied. Each vendor supporting STEP formats devotes a significant amount of effort to obtain compliance. Furthermore, the effort is spread over each of the vendors. This means dozens or hundreds of person-years of effort, as compared to just a few person-years devoted to the PIF research project.

It must also be pointed out that compliance with the STEP standard does not imply complete and error-free movement of data between vendor applications. Many problems still remain.

The representations being used by the PIF community contain are further from the implementation, and therefore require more manual effort to implement. In contrast, EXPRESS is closer to the implementation and therefore, much of the manual effort is reduced at the expense of flexibility in implementation.

5.3 Data Access via Mapped Ontologies

5.3.1 Overview

The motivation behind this scenario is the same as the last one. The key difference is that here, there is no explicit shared ontology; instead, mapping rules are used to define what a term in one ontology means in another ontology. A mediator uses these rules at runtime so that applications can access each other's data. This approach has the advantage of not requiring the application developers to explicitly agree on a shared ontology. Supporting technologies include parser generators, printers, and mediators. The principle actors are ontology authors and application developers.

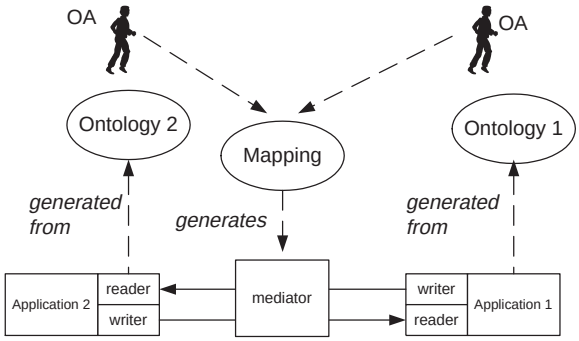


Figure 6: Data Access via Mapped Ontologies

In this scenario, each application wishing to exchange data has it's own local ontology. Application developers cooperate to create a shared mapping that relates terms in different ontologies. This mapping is used to generate a mediator, which maps operational data expressed in the terminology of one ontology into operational data expressed the other ontology.

5.3.2 Example

A developer of an application (e.g., electrical power suppliers) wants to share data with another application (e.g., schematic viewer). Each application has its own ontology created in EXPRESS. The developers agree on a mapping (e.g., represented in EXPRESS-X), which relates terms in the power supply application with electrical schematics. The mapping is used to generate a mediator that maps those portions of the electrical power supply data into schematic data.

5.3.3 Variations

Variations for data exchange via a mapped ontology are the same as for a shared ontology. Another use of mapped ontologies is to define views. One ontology represents a view of data that can be mapped to a larger ontology. This is analogous to use of database views.

5.4 Shared Services

5.4.1 Overview

The motivation behind shared services is neutrality (i.e., language, machine, operating system, location). Developers achieve this by agreeing on a shared ontology, which

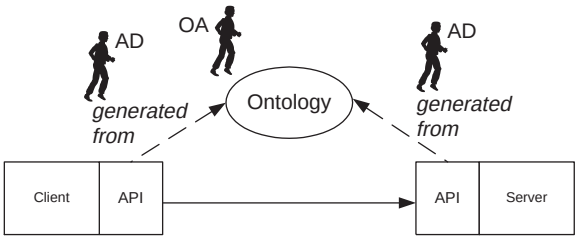


Figure 7: Shared Services

defines interfaces in multiple target languages. This is very similar to data access via shared ontologies, except for the focus of what is being shared. Supporting technologies include interface generators and marshaling routines. The principle actors are ontology authors and application developers.

In this scenario, an ontology author creates an ontology, which different application developers agree to use. Parser generators and printers are used to generate application interface definitions that each application uses to read and write data.

5.4.2 Example

An ontology author uses a language such as IDL or UML to create an ontology for objects in a domain of discourse (e.g., product data management). The ontology is used to generate interface code for the client and server (e.g., using CORBA). Client applications can then interface with services on the server regardless of location, operating system, or location.

5.4.3 Maturity

The standards and machinery supporting this approach are relatively mature. Success depends primarily on agreement on an ontology with enough semantic richness to satisfy the requirements of the client and server.

6 Scenarios: Indexing

6.1 Concept Based Search

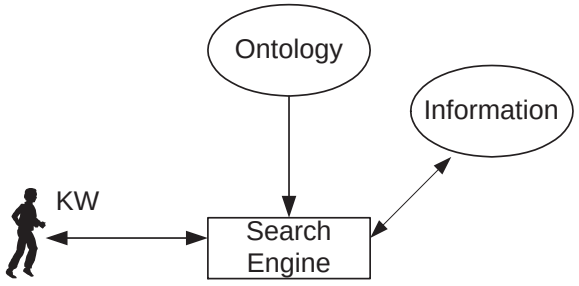


Figure 8: Concept-Based Search

6.1.1 Overview

The motivation behind concept-based search is location of artifacts (e.g., documents) in some repository. Knowledge workers accomplish this by using an ontology that a search engine applies as an index into the repository. Supporting technologies include ontology browsers and search engines. The principle actors are ontology authors and knowledge workers.

In this scenario, an ontology author creates an ontology that different knowledge workers use to identify concepts in which they are interested. The search engine uses these concepts to locate desired artifacts from a repository.[5]

6.1.2 Example

An ontology author creates an ontology (typically a simple taxonomy with relations between terms). A knowledge worker selects terms from the ontology based on concepts they are searching for in the ontology. A specialized search engine uses these terms to locate relevant documents in a repository.

6.1.3 Variations

Variations mainly deal with whether artifacts in the repository are tagged and the semantic richness of the ontology. A richer ontology can be used to make minor inferences to improve search capability.

6.1.4 Maturity

Many commercial Internet portals are beginning to explore use of concepts described in this scenario. Several research projects, more closely aligned to this idea, are being explored.

7 Discussion and Future Work

We have presented a framework for understanding ontology applications, and used it to highlight the many similarities between work being done in different areas. We intend to disseminate this framework to the STEP, OMG and information integration communities. We hope to increase the repertoire of tools and methods to the wider community for achieving their goals. It is important to emphasize that an application may integrate more than one of the scenarios presented. We hope that by bringing these all together in one place, workers may be inspired to creatively combine them to make more useful applications.

This is on going work and there is much more to be done. This includes:

[5] We have chosen to draw the figure from the KW's perspective, for whom the fact that the search engine is an ontology based application is irrelevant. It is equally valid to introduce an application developer actor who uses the ontology and to view the knowledge worker as an application user.

7.1 More Details

Many interesting variations exist for each of the above scenarios, which we have not mentioned. In addition, some of the ones that we have mentioned are important enough to warrant their separate diagrams, examples, and discussion. There is much more to be said about the maturity of each of these approaches.

We are particularly interested in illuminating why some of the same approaches seem to have great limitations in some contexts, and yet are seeing commercial success in other contexts. For example, PIF versus EXPRESS as applications of the Data Access via Shared Ontology scenario.

7.2 Alternate Technologies and Tradeoffs

For each of the areas where ontologies may be applied, we would like to have an explicit account of under what circumstances any given approach is likely to work. We would also like to identify alternate technologies, which can accomplish the same goals, as well as their tradeoffs. For example, the use of ontologies as interchange formats is an unproven technology for sharing complex operational data. The alternative is to build point to point translators. There are a whole host of unexplored issues.

Eventually, this can then be turned into guidelines for potential ontology application developers, who can be guided to what approach to use under their specific circumstances.

7.3 More areas

The following areas have not been explored sufficiently, if at all. They need to be brought into the framework.

- Ontologies used for indexing, is becoming a field of its own with major commercial use (e.g., Yahoo!) as well as a plethora of research papers published recently. It would probably be useful to have a separate framework for this area alone.
- The role of large scale general purpose ontologies such as Cyc.
- The role of natural language ontologies, such as WordNet.
- The domain modeling community within software engineering.
- Information Integration e.g., heterogeneous databases, data warehouses.

7.4 Populate the Framework

We would like to list a wide variety of actual systems reported in research and industry and classify them using this framework.

7.5 Recommend Future Research

In performing this analysis, we hope to provide a thorough review of the state of the art of ontology application. With a populated framework, and a better understanding of the maturity of various approaches, and the various tradeoffs, we hope that this will naturally suggest fruitful areas for further research.

Acknowledgements

Peter Clark, Florence Tissot, Deborah McGuinness, Richard Fikes, Doug Lenat, and Fritz Lehman provided helpful feedback during discussions on earlier versions of this paper.

References

[1] W. Grosso, J. Gennari, R. Fergeson, and M. Musen. When knowledge models collide (how it happens and what to do). In *Proceedings of the Eleventh Workshop on Knowledge Acquisition, Modeling and Management. Track: Shareable and reusable components for knowledge systems*, Banff, Alberta, Canada, April 1998. See URL: ksi.cpsc.ucalgary.ca/KAW/KAW98/KAW98Proc. html.

[2] T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[3] T. Gruber and G. Olsen. An ontology for engineering mathematics. In *Proc. of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kauffman, 1994. Also available as Stanford Knowledge Systems Laboratory technical report KSL-94-18.

[4] M. Gruninger and M.S. Fox. The logic of enterprise modelling. In J. Brown and D. O'Sullivan, editors, *Reengineering the Enterprise*, pages 83–98. Chapman and Hall, 1995.

[5] I. Jacobson, M. Christerson, P. Jonsson, and Gunnar Overgaard. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, Wokingham, England, 1992.

[6] P.D. Karp, M. Riley, S.M. Paley, and A. Pelligrini-Toole. Ecocyc: encyclopedia of e.coli genes and metabolism. *Nucleic Acids Res.*, 24:32–40, 1996. See URL: ecocyc.panbio.com/ pkarp/mimbd/94/abstracts/pkarp. html.

[7] J. Lee, G. Yost, and PIF Working Group. The pif process interchange format and framework. Technical Report 180, MIT Center for Coordination Science, 1995.

[8] D. McGuinness. Ontological issues for knowledge-enhanced search. In N. Guarino, editor, *Formal Ontology in Information Systems*, pages 302–316, Trento, Italy, June 1998.

[9] Workflow Management Coalition Members. Glossary - a workflow management coalition specification. Technical report, The Workflow Management Coalition, 1994.

[10] International Standards Organization. *The EXPRESS Language Reference Manual*. 1994. Reference No: ISO 10303-11:1994(E).

[11] D. Schenck and P. Wilson. *Information Modeling the EXPRESS Way*. Oxford University Press, 1994.

[12] M. Uschold and M. Gruninger. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11(2), 1996. Also available as AIAI-TR-191 from AIAI, The University of Edinburgh.

[13] M. Uschold, M. Healy, K. Willamson, P. Clark, and S Woods. Ontology reuse and application. In N. Guarino, editor, *Formal Ontology in Information Systems*, pages 179–192, Trento, Italy, June 1998.

[14] M. Uschold, M. King, S. Moralee, and Y. Zorgios. The enterprise ontology. *Knowledge Engineering Review*, 13(1), 1998. Also available as AIAI-TR-195 from AIAI, The University of Edinburgh. This ontology was developed as part of the Enterprise Project, see *http://www.aiai.ed.ac.uk/∼entprise/enterprise/* for further information.

[15] M. (editor) Uschold. Knowledge level modelling: Concepts and terminology. *Knowledge Engineering Review*, 13(1), 1998. Also available as AIAI-TR-196 from AIAI, The University of Edinburgh.

[16] A. Valente, T. Russ, R. MacGregor, and W. Swartout. Building and (re)using an ontology of air campaign planning. *IEEE Intelligent Systems*, 14(1):27–36, January/February 1999.

[17] R. Waldinger, Y.V. Srinivas, A. Goldberg, and R. Jullig. *Specware Language Manual*, 1996.

[18] K. Williamson, M. Healy, and R. Jasper. Formally specifying engineering design rationale. Technical Report ISSTECH-97-011, Applied Research and Technology, The Boeing Company, 1997.