

A Framework for Understanding and Classifying Ontology Applications

Robert Jasper and Mike Uschold
robert.j.jasper@boeing.com
michael.f.uschold@boeing.com

Boeing Math and Computing Technology, P.O. Box 3707, Seattle, USA

Abstract

For¹ ontologies to be cost-effectively deployed, we require a clear understanding of the various ways that ontologies are being used today. To achieve this end, we present a framework for understanding and classifying ontology applications. We identify four main categories of ontology applications: 1) neutral authoring, 2) ontology as specification, 3) common access to information, and 4) ontology-based 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, who the principal actors are and what they do. We illuminate the similarities and differences between scenarios.

We draw on work from other communities, such as software developers and standards organizations. We use a relatively broad definition of ‘ontology’, to show that much of the work being done by those communities may be viewed as practical applications of ontologies. The common thread is the need for sharing the meaning of terms in a given domain, which is a central role of ontologies. An additional aim of this paper is to draw attention to common goals and supporting technologies of these relatively distinct communities to facilitate closer cooperation and faster progress².

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, (McGuinness, 1998; Guarino *et al.*, 1999). And yet, outside the AI ontology community, industry has been regularly using ontologies successfully (even though they may not call them ontologies).

In order to cost-effectively deploy ontologies, we require a clearer understanding of the myriad of situations and contexts and ways that ontologies are being applied today. To that end, we present a framework for understanding and classifying ontology applications. The main focus of this paper is to describe the framework and to immediately increase our understanding of ontology applications. We use examples to illustrate the framework. In doing so, we lay the foundation for a comprehensive classification of ontology applications which is beyond the scope of this paper. In the longer term, we hope this work will lead to the development of specific guidelines for how to use ontologies to achieve specific benefits under a given set of circumstances.

We draw on similar work being done outside the ontology research community. Other communities include: software developers and standards organizations as well as the database community, who are working to integrate heterogeneous databases. 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

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

²This paper is a revised version of a paper by the same title that appeared in the proceedings of the IJCAI-99 ontology workshop. There are also minor differences in emphasis reflecting the different the different goals of the KAW workshop.

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.

Another goal of this paper is to draw attention to the similarities among the different communities, to facilitate faster progress for everyone. These groups strive to overcome the barriers noted in the previous paragraph; ironically, the same underlying issues also create barriers to closer cooperation between the different communities who are addressing similar problems. We hope to lower these barriers by identifying and highlighting the commonality among these communities, and pointing out important differences. 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.

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 (Uschold, 1998) (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.”

This broad interpretation helps to show how both the goals and the technical approaches 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 (Gruber, 1993), EXPRESS³ (Schenck & Wilson, 1994; International Standards Organization, 1994) and IDL) and translation tools. Thus, we can easily view a number of standardization efforts (e.g., STEP⁴, OMG⁵) as practical applications of ontologies.

2 Overview of the Framework

The main part of the framework consists of a set of *ontology application scenarios*. By ‘application’, we mean a system or process, (usually a computer program) that makes use of or benefits from the ontology. A scenario is an abstract use case for a class of similar applications. It describes a particular situation in which an ontology is put to use for some specific purpose(s). We identify four main categories of ontology applications: 1) neutral authoring, 2) ontology as specification, 3) common access to information, and 4) ontology-based 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.

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 and 5) the maturity level.

³an object-flavored language for specifying information models, originally developed as part of the STEP standard.

⁴Standard for the Exchange of Product model data; an informal name for the ISO-10303 family of standards

⁵Object Management Group; see www.omg.org

Two additional distinctions that play an important role in some scenarios, are: 1) how meaning of terms is represented (e.g., informal or formal) and 2) sharing vs exchange (e.g., pass by reference or pass by value). In the remainder of this section, we describe the key dimensions and distinctions 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 (Uschold, 1998).

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.⁶ 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 (Schenck & Wilson, 1994; International Standards Organization, 1994) which is at level L_1 . Similar analogies exist for object sharing and exchange standards (e.g., OMG).

Information at the same level can be 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.

Terms may also 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.

(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.

⁶Some 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 .

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: 1) Ontology representation languages-(e.g., UML, Express, Ontolingua, XML); 2) Knowledge interchange languages: (e.g., KIF, PIF(Lee *et al.*, 1995), CDIF); 3) Translation tools: (e.g., Ontolingua translators, CDIF-tools, StepTools, ... (lots!)); and 4) Distributed Objects: (e.g., CORBA⁷, 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 vary 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 taxonomy 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 (Gruninger & Fox, 1995) and PIF (Lee *et al.*, 1995).

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.

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 present a more challenging task but may promise greater rewards. In the short term, light-weight ontologies (less rich in meaning) can be easier to apply in working systems. An excellent example of this is the multiplicity of uses of ontologies used to semantically structure a information repository as a basis for search and retrieval. This is happening 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.

The nature of the two tasks, search vs inter-operation are dramatically different. A light-weight ontology will not be much help in developing translation tools, where it can be of immediate benefit for search purposes. However, even in

⁷Common Object Request Broker Architecture

the search area, there are obvious benefits of richer ontologies. For example, they can support inference which could be used to increase both precision and recall in a semantically well structured information repository.

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.

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. They are analogous to Ivor Jacobson's use cases (Jacobson *et al.*, 1992). 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 one or more concrete examples. Where appropriate, we identify a number of alternate variations of the main scenario. Finally, we assess the maturity level of each scenario.

Given the diverse applications of ontologies from the literature, and the various dimensions by which they can be classified, it was not clear how best to group them into major categories. Eventually, four main categories emerged.

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.

Ontology as Specification: An ontology of a given domain is created and used as a basis for specification and development of some software. Benefits of this approach include documentation, maintenance, reliability and knowledge (re)use.

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.

Ontology-Based Search: An ontology is used for searching an information repository for desired resources (e.g. documents, web pages, names of experts). 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 Scenario: Neutral Authoring

The basic idea of these scenarios is to author an artifact in a single language, and to have that artifact translated into a different format for use in multiple target applications. The benefits of this approach include decreased cost of reuse and portability of knowledge across applications, improved application maintainability (because an artifact is only authored in one place, and can be centrally updated) and long term knowledge retention (e.g., via reduced disruption from changes in vendor formats).

An important distinction is whether the authored artifact requiring translation is an ontology, or operational data. These are discussed in the next two sections.

4.1 Authoring Ontologies

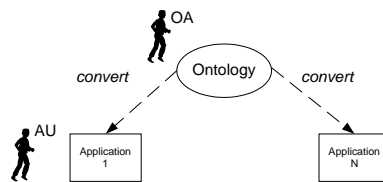


FIGURE 1: Authoring Ontologies

4.1.1 Overview

The motivation behind authoring neutral ontologies is decreased cost of reuse and maintenance of knowledge. To accomplish this, the actors develop an ontology, which is translated and used in multiple operational target systems. Supporting technologies include unidirectional ontology translators. The principle actors are the ontology author and application user.

In this scenario, the ontology author creates an ontology, which is translated into different formats and used in multiple target applications. For example, the ontology may become part of the knowledge base used in the application. Application users then interact with an operational system to perform their desired tasks.

4.1.2 Examples

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 § 6, 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.

4.1.3 Maturity

Totally automated translation of ontologies into operational targets has been difficult and typically relies on translation of idiomatic expressions (Gruber, 1993). For case studies and analysis of some of these problems see (Grosso *et al.*, 1998; Uschold *et al.*, 1998; Valente *et al.*, 1999).

4.2 Authoring Operational Data

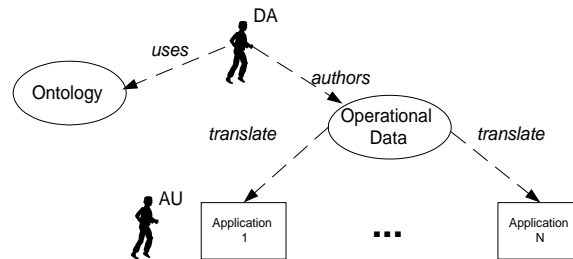


FIGURE 2: Authoring Operational Data

Neutral authoring of operational data is identical in structure to neutral ontology authoring. The focus is on authoring and translating operational data rather than an ontology. 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 [operational data] 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 (Gruber & Olsen, 1994) 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 outlined in (Uschold & Gruninger, 1996).

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 axiomatizations, then there is scope for partially automating the construction of the translators and/or in maintaining them when there are changes in either 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 is addressed in detail in § 6. The next scenario is similar to neutral authoring, but with some important differences.

5 Scenario: Ontology as Specification

The basic idea of this scenario is to author an ontology which models the application domain, and provides a vocabulary for specifying the requirements for one or more target applications. The richer the ontology is in expressing meaning, the less the potential for ambiguity in creating requirements. The software is based on the ontology, which thus plays an important role in the development of the software. The benefits of this approach include documentation, maintenance, reliability and knowledge (re)use.

Structurally, this scenario is similar to Neutral Authoring (see figure 3). In both cases, the ontology plays an important role in the development of the application. There are two important differences. First, the ontology is useful in this scenario even if there is only one application. Second, unlike the neutral authoring scenario, the ontology is not translated, per se. Rather, it guides development of the target application(s).

In both scenarios, the target application is intimately related to and in some sense ‘contains’ the ontology. In the neutral authoring scenario, the ontology is an explicit component of the application. Not so in the current scenario. Every

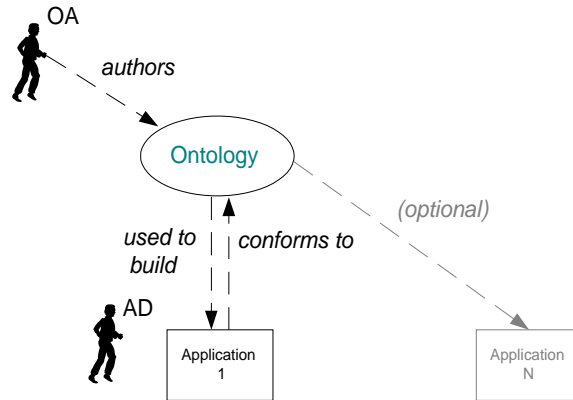


FIGURE 3: Ontology as Specification

application ‘has’ an ontology, whether it is explicit or implicit. If a principled methodology for software development is used, e.g., by building a conceptual model of the domain in UML, then this will comprise the explicit ontology for the application. Otherwise, the ontology of a software application may only be implicit. In the ontology as application scenario, the ontology of the target application is not only explicit, but it is used to develop the software.

5.0.6 Examples

KADS / CML – 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, documentation is improved because there is an explicit representation of the ontology that the software is based on. Better documentation always makes maintenance easier. Reuse is achieved if the ontology is used for different applications in the same domain.

Protoge-II – Ontologies are also used to help automate the process of knowledge acquisition, and software development. Protoge-II (Rothenfluh *et al.*, 1996), is a system which automatically generates knowledge acquisition tools from an ontology. This ensures a tight connection between the ontology and the user interface, which facilitates the entering of the knowledge. The resulting knowledge bases are used in the target application(s).

Documentation is improved because there is an explicit connection between the ontology and the target software. This in turn, assists in maintenance. There is also a more direct benefit in terms of maintenance because if the ontology changes, it is much easier to update the knowledge acquisition tools because they are automatically created. There is also scope for automating the migration of existing knowledge bases when the underlying ontology changes.

Reuse is achieved because the ontology can be used in many different knowledge based systems.

Information Modeling – Data modeling languages such as EXPRESS (Schenck & Wilson, 1994; International Standards Organization, 1994) and IDEF1X (Bruce, 1992) are used to create data base schema, which can be thought of as ontologies. In the case of EXPRESS, The data base schema specifies the physical file format for a target implementation. More generally, information and system modeling languages like UML can also be used to create ontologies which serve to specify target software.

Object-Oriented Modeling – A special case of information modeling is object-oriented analysis and design, which has become popular in recent years. The analysis phase of this approach to building software consists of identifying a taxonomy of classes which provides the vocabulary for specifying the software requirements. This taxonomy, along with the attributes (or slots) for each is essentially the same thing as the ontologies produced in the examples noted above. A primary motivation for the object-oriented methodology is reuse of methods in particular, but also of attributes through class inheritance. This kind of reuse is analogous to reusing an existing ontology when building other systems, or larger ontologies. However, it is a different form the notion of using an ontology to achieve knowledge sharing via neutral authoring, or as a neutral interchange format.

Software Synthesis – Automated software synthesis into multiple target languages (e.g., using Specware (Waldinger *et al.*, 1996; Williamson *et al.*, 1997)) 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.

5.0.7 Maturity

The KADS methodology is widely used for building knowledge based systems. However there is no formal connection between the ontology and the developed software. The main role of the ontology is to guide the development of the software, and it serves the important purpose of documenting the system. We stress however, that unless the ontology is kept in synch with the application, then the utility of the ontology as a documentation aid will be minimal. In practice, this is all too frequently what happens.

In the Protoge II system, the process of using the ontology to develop the software is partially automated, and there is a much tighter connection between the ontology and the implemented software. This system has been used to produce fielded applications with dozens of licensed users, mostly in the medical domain.

For Express, IDEF1X, UML and various other such languages, there are commercial tools for automatically generating parts of the target software from the specifications. There are also various tools for translating between these languages (e.g. Express and UML).

Semi-automated software synthesis with a formal connection between the ontology (qua specification) and the software shows some promise, but it has not been a primary focus of the ontology community.

6 Scenario: 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.

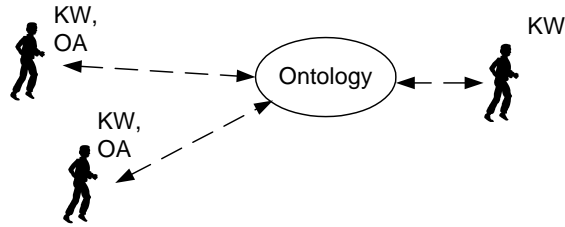


FIGURE 4: Human Communication

6.1 Human Communication

6.1.1 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. this is the only scenario in this paper which does not require a computer.

6.1.2 Examples

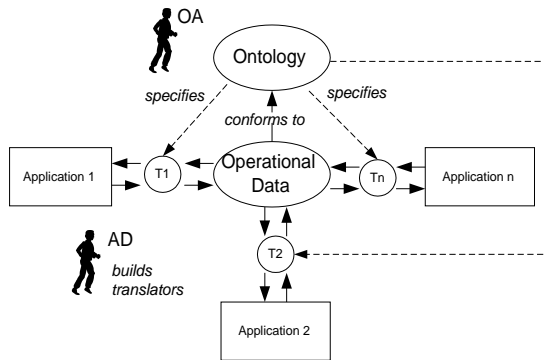
A glossary of terms to enable different working groups, who may have different jargon, to understand each other—(e.g., the workflow management coalition reference document(Members, 1994)). Producing glossaries, and providing common access for humans to important knowledge assets is a key focus of the 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.

6.1.3 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.

6.1.4 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 assistance in creating these ontologies, however, we are not aware of them.



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

FIGURE 5: Data Access via Shared Ontology

6.2 Data Access via Shared Ontology

6.2.1 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. Translators are [almost?] always manually created, although in some cases, the API for reading and writing from/to a new format may be automatically generated using parser generators and printers (see variation below). This helps, but the bulk of the work is deciding what calls to the API are required to translate a given data item stored internal data structures – this remains a challenging manual effort. Inter-operation between the multiple applications is made possible by allowing them to access the same information.

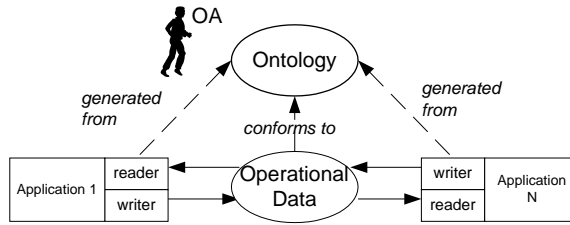
6.2.2 Examples

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 (Karp *et al.*, 1996) 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.

6.2.3 Variations

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



In this variation translation between formats is achieved using readers and writers which reside in the applications.

FIGURE 6: Data Access via Shared Ontology: variation

application are converted into the target format using readers and writers internal to the application. So there is no explicit language to language translation per se, but the effect is the same – there is 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 of 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.

6.2.4 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 sheer 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.

Although EXPRESS may be used to state important constraints as rules, these rules are not themselves shared or exchanged and thus do not have to be translated. Instead, they are used to maintain integrity of the data. In the PIF effort, the specification of operational behaviour is intended to be shared. Fully general techniques for translating behavioural specifications is beyond the current state of the art. Although not being addressed at this time, this is included in the long term vision for STEP.

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.

6.3 Data Access via Mapped Ontologies

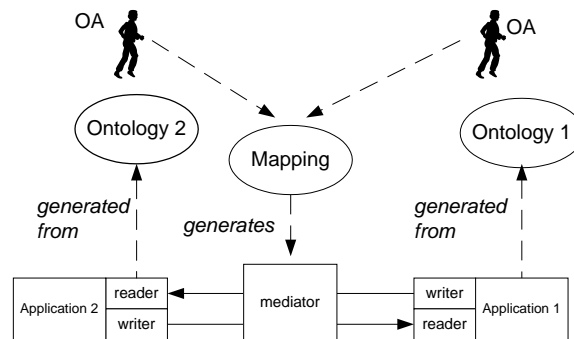


FIGURE 7: Data Access via Mapped Ontologies

6.3.1 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.

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.

6.3.2 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.

6.3.3 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.

6.4 Shared Services

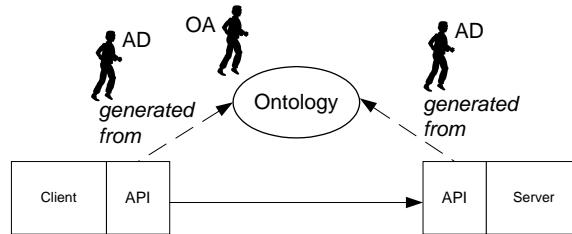


FIGURE 8: Shared Services

6.4.1 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 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.

6.4.2 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.

6.4.3 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.

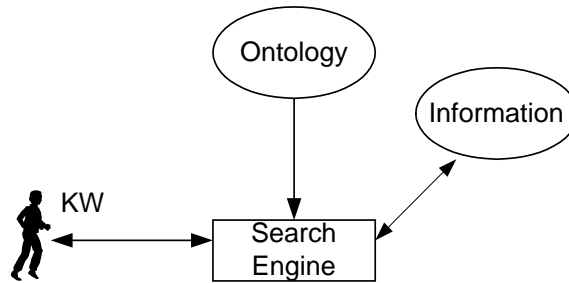


FIGURE 9: Ontology-Based Search

7 Scenario: Ontology-Based Search

The basic idea of this approach is to use an ontology for searching an information repository for desired resources (e.g. documents, web pages, names of experts). The motivation is to improve precision and/or recall as well as reduce the overall amount of time spent searching (see figure 7).

The principle actors are knowledge workers (i.e., application users) and ontology authors. Supporting technologies include ontology browsers, search engines, automatic tagging tools, automatic classification of documents, natural language processing, meta-data languages (e.g., XML), natural language ontologies, large general-purpose knowledge bases and thesauri, and knowledge representation and inference systems.

In this scenario, an ontology author creates an ontology that assists knowledge workers in identifying concepts that they are interested in. The search engine uses these concepts to locate desired resources from a repository.⁸

7.1 Examples

Knowledge-Based Discovery Tool (Eilerts *et al.*, 1999) A knowledge worker enters words related to the concept of interest (e.g. tank). The browser consults a large WordNet-based ontology, identifies any potential ambiguities, and asks the user to identify the correct concept (e.g. a storage tank, or an army tank). The browser re-formulates the query by automatically appending the negation of terms closely related to the wrong concept. For example, if the desired concept is storage tank, then responses which contain words strongly associated with 'army' are excluded.

Yahoo! Another approach is to use the ontology as an index into the repository. This requires that all items in the repository are linked to items from the index. An ontology author creates an ontology (typically a simple taxonomy with relations between terms). A knowledge worker uses this taxonomy to help limit the scope of their search. A specialized search engine uses these terms to locate relevant documents in a repository.

7.2 Variations

There are various specific roles that an ontology may play to assist search. These may be used separately, or in concert.

- as a basis for semantically structuring and organizing the information repository; this structure may be used in at least two distinct ways:

⁸We 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.

- as a conceptual framework to help the user think about the information repository and formulate queries (e.g. the Yahoo! taxonomy)
- as a vocabulary for meta-data used for indexing and/or tagging the documents in the repository (this does not require user interaction)
- to assist in query formulation; here also, the ontology can be used in at least two different ways, depending on the involvement of the user. The ontology can be used to:
 - drive the user interface for creating and refining queries, for example, by disambiguating (this corresponds to example 1 above).
 - perform inference to improve the query (not requiring user interaction)

In addition to the specific role the ontology plays, scenarios for using ontologies for search vary according to:

- whether the queries are strictly word-based, or whether semantic concepts are used;
- whether the documents are explicitly tagged with concepts from the ontology, or not;
- whether and how various steps can be partially or fully automated.

Automation may help in various ways. One important area of research is the automatic tagging and classification of documents. Another is the use of natural language processing to kick-start the ontology development process by automatically identifying potential concepts and producing a preliminary taxonomy. Natural language processing may also be used to help understand the user's query. A very challenging and active area of research is in using the ontology as a basis for performing inference to improve search. Subsumption can be helpful ((McGuinness, 1998)), and is provided with description logic based languages. The Boeing Company has an internal prototype search tool which performs inference using various concept associations from a 35,000 term thesaurus to improve search queries. The semantic richness of the ontology may be an important factor here. We hypothesize that a richer ontology can improve inference capability, which can in turn, improve search.

7.3 Maturity

Many commercial Internet portals are beginning to explore the use of concepts and technologies described above for ontology-based search. Several research projects, more closely aligned to this idea, are being explored.

8 Discussion and Future Work

We have presented a framework for understanding ontology applications. One of the reasons for the diversity of ontology applications, is that there are many different notions about what an ontology is. This partly correlates to the different applications, but it is the main point of this paper 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:

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.

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 advised about what approach to use in their specific circumstances.

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

- The role of large scale general purpose ontologies such as Cyc.
- The domain modeling community within software engineering.
- Information Integration e.g., heterogeneous databases, data warehouses.

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.

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 and John Thompson helped identify the distinctions for the ontology-based search scenarios. Peter Clark, Florence Tissot, Deborah McGuinness, Richard Fikes, Doug Lenat, and Fritz Lehman provided helpful feedback during discussions on earlier versions of this paper. Helpful feedback was also provided by several anonymous referees.

References

- Bruce, T. (1992). *Designing Quality Databases with IDEF1X Information Models*. Dorset House Publishing.
- Eilerts, E., Lossau, K., & York, C. (1999). Knowledge base discovery tool. In *Proceedings of AAAI-99*.
- Grosso, W., Gennari, J., Fergeson, R., & Musen, M. (1998). When knowledge models collide (how it happens and what to do). In *Proceedings of the Eleventh Workshop on Knowledge Acquisition, Modeling and Manage-*

ment. Track: *Shareable and reusable components for knowledge systems*, Banff, Alberta, Canada. See URL: <http://ksi.cpsc.ucalgary.ca/KAW/KAW98/KAW98Proc.html>.

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

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

Gruninger, M. & Fox, M. (1995). The logic of enterprise modelling. In Brown, J. & O'Sullivan, D., (Eds.), *Reengineering the Enterprise*, pages 83–98. Chapman and Hall.

Guarino, N., Masolo, C., & Vetere, G. (1999). Ontoseek: Using large linguistic ontologies for accessing on-line yellow pages and product catalogs. *IEEE Intelligent Systems*, 14(3):70–80.

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

Jacobson, I., Christerson, M., Jonsson, P., & Overgaard, G. (1992). *Object-Oriented Software Engineering: A Use Case Driven Approach*. Wokingham, England, Addison-Wesley.

Karp, P., Riley, M., Paley, S., & Pelligrini-Toole, A. (1996). Ecocyc: encyclopedia of e.coli genes and metabolism. *Nucleic Acids Res.*, 24:32–40. See also: <http://ecocyc.panbio.com/pkarp/mimbd/94/abstracts/pkarp.html>.

Lee, J., Yost, G., & Group, P. W. (1995). The pif process interchange format and framework. Technical Report 180, MIT Center for Coordination Science.

McGuinness, D. (1998). Ontological issues for knowledge-enhanced search. In Guarino, N., (Ed.), *Formal Ontology in Information Systems*, pages 302–316, Trento, Italy.

Members, W. M. C. (1994). Glossary - a workflow management coalition specification. Technical report, The Workflow Management Coalition.

Rothenfluhh, T., Gennari, J., Eriksson, H., Puerta, A., Tu, S., & Musen, M. (1996). Reusable ontologies, knowledge-acquisition tools, and performance systems: Protoge-ii solutions to sisyphus-2. *International Journal of Human-Computer Studies*, 3-4(44):303–332.

Schenck, D. & Wilson, P. (1994). *Information Modeling the EXPRESS Way*. Oxford University Press.

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

Uschold, M., Healy, M., Williamson, K., Clark, P., & Woods, S. (1998). Ontology reuse and application. In Guarino, N., (Ed.), *Formal Ontology in Information Systems*, pages 179–192, Trento, Italy.

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

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

Waldinger, R., Srinivas, Y., Goldberg, A., & Jullig, R. (1996). *Specware Language Manual*.

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